

National College of Ireland  
BSc in Computing (Mobile Stream)  
May 2017

Ronan Browne  
X13114522  
[X13114522@student.ncirl.ie](mailto:X13114522@student.ncirl.ie)

Crowdsourcing a solution to bike theft in Dublin City  
alongside theft prevention via remote sensors.

Technical Report for Bike Pro App



## Contents

Executive Summary.....	4
1 Introduction.....	7
1.1 Background .....	7
1.2 Aims .....	7
1.3 Technologies .....	7
2 System .....	9
2.1 Requirements .....	9
2.1.1 Functional requirements .....	9
2.1.2 Use Case Diagram.....	11
2.1.3 Requirement: Register a user .....	12
2.1.4 Requirement : Login.....	14
2.1.5 Requirement : Register a bike.....	16
2.1.6 Requirement : Edit a Bike / Register as Stolen .....	18
2.1.7 Requirement : Viewing Stolen Bike Database.....	21
2.1.8 Requirement : Reporting a sighting.....	23
2.1.9 Requirement : Google Maps Integration. ....	26
2.1.10 Requirement: Bluetooth Sensor Integration. ....	29
2.1.11 Requirement: Scanning for stolen bikes using Bluetooth sensors and users device. ....	31
2.2 Data requirements.....	34
2.2.1 Database Design .....	34
2.2.2 Data Conversions .....	37
2.3 User requirements.....	38
2.4 Environmental requirements.....	38
2.5 Usability requirements .....	39
2.6 Design and Architecture .....	43
2.6.1 Architecture Design.....	43
2.6.2 Software Architecture.....	45
2.6.3 Hardware Architecture .....	49
2.6.4 Security Architecture.....	51
2.6.5 Communication Architecture .....	53

2.6.6	Application Program Interfaces .....	54
2.6.7	User Interface Design .....	55
2.7	Implementation .....	59
2.8	Graphical User Interface (GUI) Layout .....	67
2.9	Testing.....	90
2.9.1	Technical Testing.....	90
2.9.2	Non-Technical testing .....	98
	Acceptance Testing:.....	98
	Five Second Test: .....	98
	Trunk Test .....	99
2.10	Evaluation.....	100
2.11	Performance.....	102
3	Conclusions.....	106
4	Further development or research .....	107
5	References .....	108
6	Appendix .....	110
6.1	Key Terms .....	110
6.2	Project progress tracked through GIT.....	111
6.3	Project Proposal .....	112
6.3.1	Objectives .....	114
6.3.2	Background.....	114
6.3.3	Approach.....	115
6.3.4	Project Plan.....	116
6.3.5	Technical Details.....	117
6.3.6	Evaluation .....	118
6.3.7	References.....	118
6.4	Monthly Journals .....	120
6.4.1	Reflective Journal Month 1.....	120
6.4.2	Reflective Journal Month 2.....	121
6.4.3	Reflective Journal Month 3.....	123
6.4.4	Reflective Journal Month 4.....	125
6.4.5	Reflective Journal Month 5.....	128

6.4.6	Reflective Journal Month 6.....	130
6.4.7	Reflective Journal Month 7.....	131

## Declaration Cover Sheet for Project Submission

### SECTION 1 *Student to complete*

<b>Name:</b>	<b>Ronan Browne</b>
<b>Student ID:</b>	<b>X13114522</b>
<b>Supervisor:</b>	<b>Dr. Paul Hayes</b>

### SECTION 2 Confirmation of Authorship

*The acceptance of your work is subject to your signature on the following declaration:*

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Ronan Browne

Date: 09/05/2017

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

# Executive Summary

## Problem addressed?

An average of 14 bikes stolen every day in Dublin (Dublin Cycling Campaign) this is a real issue for cyclists. More often than not the bike is never seen again.

When a person has their bike stolen they will most likely report it to the police, An officer will file a report and for the majority of cases that is where the issue will end.

Police are too busy with what they deem to be more serious crime to invest much time in a bike theft.

Therefore, broader thinking is required to address this issue. My solution is to leverage the growing trends of crowd sourcing and mobile applications. Crowd sourcing the bike riding community to help one another find stolen bikes.

Initially this project will be focused on bikes but further iterations of the application could easily be expanded to include other high value items such as cars.

Three key feature of the application will be

1. Notifying a user if there is unauthorized movement of their bike. This will take place through an external sensor placed on the bike which will communicate with the user's phone
2. Being able to browse a shared database of bikes listed as stolen by users and having the ability to report a suspected sighting of a stolen bike, which will notify the original owner and relevant authority's.
3. Being able to scan the immediate area for bikes that have been registered as stolen and are using the sensors. Should bikes listed as stolen be found by the system all relevant information will be displayed to the user along with an approximate distance to the stolen bike. This is triangulated via the signal strength of the Bluetooth sensor on the bike. The user can then contact the original owner through the app and let them know they have a legitimate sighting of their stolen bike confirmed through the sensor technology.

The technology used to implement this solution will include an application built on the android platform, using googles cloud hosted firebase database, google maps API integration and remote sensors.

These sensors contain a micro-computer powered by an ARM chip. The sensors include motion and temperature scanners along with NFC compatibility. The sensors transmit signals through a 2.4GHz frequency of Bluetooth with a range of up to 70m.

The application will facilitate users to help one another locate their stolen bikes. Also provide a means to prevent theft occurring via the remote sensors.

# **1 Introduction**

## **1.1 Background**

I came up with the idea for this application after my own bike was stolen in Dublin City center and I had wished there were more tools available to me to prevent this from happening or to allow me to recover my bike. Over the course of the following weeks and months I started thinking about how I could mold my experience into a software solution that could help others. These ideas formed the basis of the application discussed in this document; Bike Pro.

## **1.2 Aims**

To provide a tool which will help prevent bike theft, identify theft hot spots and give users who have had their bike stolen an extra means of being reunited with their bike.

This project will see the development of a scalable, community driven anti-theft focused mobile application. This should be a streamlined responsive application. Scalability will also be kept in mind regarding the product this application is aimed to protect.

This application will be community driven with a strong emphasis on user interaction, the users will populate the application data.

This application shall also make use of sensors which can be placed on or embedded in the bike. These sensors will communicate with the application, and monitor the bike for any unauthorized movement when parked nearby.

Any phone using the application should be able to operate as a scanning device for stolen bikes in the immediate vicinity which have the sensor technology embedded. It will be possible to triangulate the exact distance a user is from the stolen bike based on the signal strength that the bikes sensor is emitting.

## **1.3 Technologies**

### **Android (Google)**

This application shall be developed (IS DEPLPOYED) on the Android platform.

**Firestore (Google)**

Googles firestore shall be used as a cloud based database for this project. It provides the framework for a NoSQL JSON database.

**External Sensors (Estimote)**

Bluetooth enabled smart beacons will be used as a positioning device on the bike. The sensors I have chosen to use are produced by the company Estimote. These Smart sensors contain a micro-computer powered by an ARM chip. The sensors include motion and temperature scanners along with NFC compatibility. The sensors transmit signals through a 2.4GHz frequency of Bluetooth with a range of up to 70m. These signals can be picked up by and read by a user's phone. Estimote have developed an Android SDK to programmatically link the sensors with my application. I will use the Estimote SDK to programmatically link my application and the sensors



## 2 System

### 2.1 Requirements

#### 2.1.1 Functional requirements

In this section an overview of the functional requirements is provided followed by a breakdown of each requirement into its use cases.

1. Register new user:  
The application will allow a user to create a unique login ID and password which shall unique identify them to the system and allow them to login.
2. Login / Logout:  
Once successfully logged in a user may logout at any time. This will end the current session returning the application to the sign in screen.
3. View my bike:  
A user may view all bikes registered to the in a scrollable list view with a summary of that bikes details.
4. Register a bike:  
The application will allow a unique user to register their bike or bikes, providing any unique identifying features along with an image. Upon successful registration a user's bike data will be stored remotely as JSON Data.
5. Edit a Bike / Register as Stolen:  
A user may select from a list of their already registered bikes to edit a specific bike. All original information provided may be edited along with signalling that the bike has been stolen and entering its last known location.
6. Upload to Stolen Database:  
Once a user has edited a bike to indicate that it has been stolen it is then added to a new Node in our JSON database specifically for stolen bikes.
7. Viewing Stolen Bike Database:  
Any user of the application can see all bikes registered as stolen by navigating to the "Stolen bikes" menu item with in the application navigation system. This will provide a user friendly list displaying each bike registered as stolen including a picture and the details of each bike.

8. Reporting a sighting:

Any user may report if they think they have seen a bicycle that has been reported as stolen. This will send a mail to the original user where they can specify details of sighting.

9. Google maps integration:

If a User has their bike stolen they can drop a pin on a map using the google maps API this location is then saved and available for all users to see. Thus over time as bikes are stolen in the city a map of bike theft hot spots will build up providing useful data analytics to Users.

10. Bluetooth Sensor Integration:

Users have the option of attaching a Bluetooth sensor to their bike which will alert them of unauthorized movement of bike. This is designed to be used when a bike is located a short distance away from you such as outside work or college and you will be notified via smartphone that your device is being moved without your permission.

11. Scan for stolen bikes:

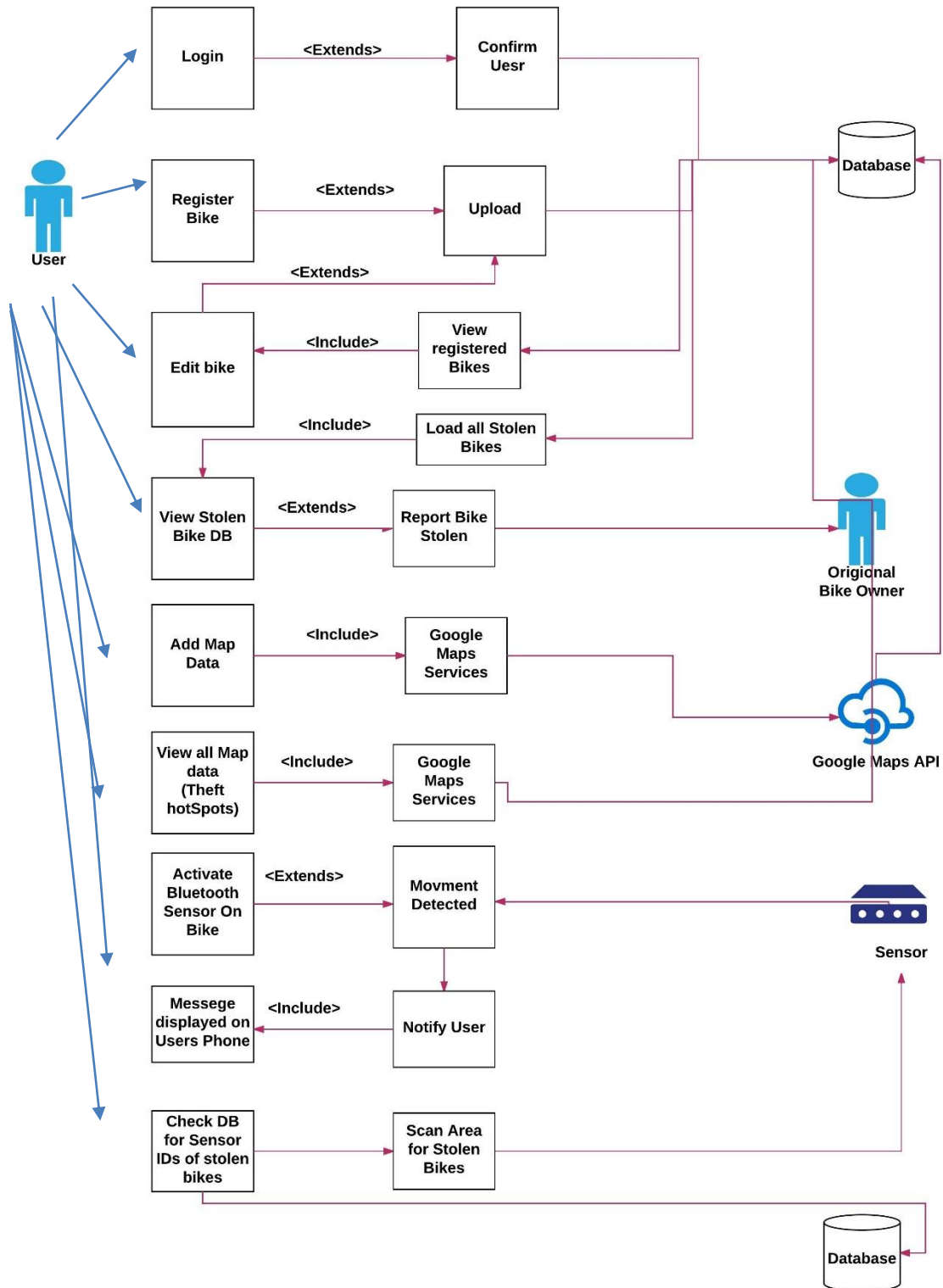
Users may scan the area for bikes that are using the system, if a device detects one of these sensors within range it will check if the unique ID of the sensor matches that of any bikes listed as stolen. If there is a match the bike details will be displayed to the user along with an approximate distance to the bike calculated based on the sensors signal strength. This distance field will update in real time as a user moves in the physical world this they can pinpoint the exact location of the bike and then report the confirmed sighting to the user through the application.

12. Home Screen:

During the development process I felt there was a need for a specify home screen with a system summary. Previously the user had just been taken to the register bike page upon launch of the application.

## 2.1.2 Use Case Diagram

The Use Case Diagram provides an overview of all functional requirements. This is a full system use case; individual use cases diagrams are given thereafter.



## 2.1.3 Requirement: Register a user

### 2.1.3.1 Description & Priority

Once the application has been downloaded and installed on a user's smartphone the first screen a user will be presented with is one prompting them to registering as a new user or login

### 2.1.3.2 Use Case 1

#### Scope

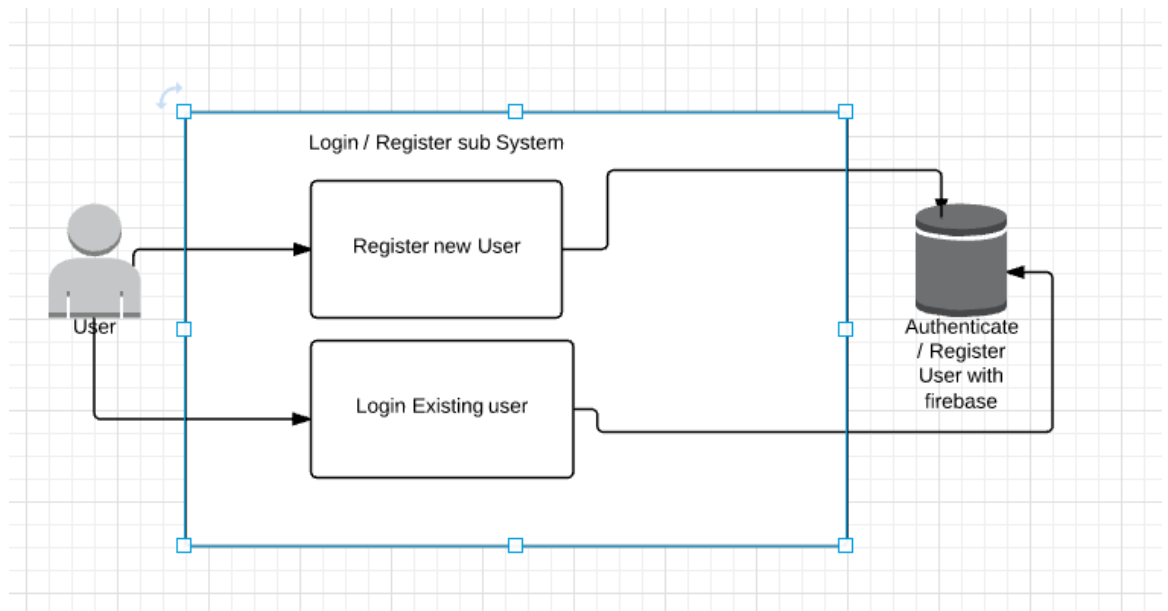
User interaction with Android application

#### Actors

1. User
2. Database

#### Use Case Diagram

\*diagram shows log in and register use case



## Flow Description

### Precondition

The application must be installed and a user registered.

### Activation

Active upon launching of application by a registered user.

### Main flow

1. User Launches app
2. User enters username and password
3. User clicks create account
4. The system saves these details to Firebase Database.

### Exceptional flow

The user exits the app before they have clicked create user.

### Termination

Application closed by user.

### Post condition

The system goes into a wait state until user interaction is detected.

```
CreateUser(String email, String password){
    FirebaseAPI.createNewUser(email,password){
    {
        If(Successful)
            SignInUser()
        else
            DisplayAppropriateFailureMessage()
    }
}
```

**Create new user sudo code from SignIn.java class**

### 2.1.4 Requirement : Login

#### 2.1.4.1 Description & Priority

A user may login to the app with a pre-existing account. Once successfully logged in a user may logout at any time. This will end the current session returning the application to the sign in screen.

#### 2.1.4.2 Use Case 2

##### Scope

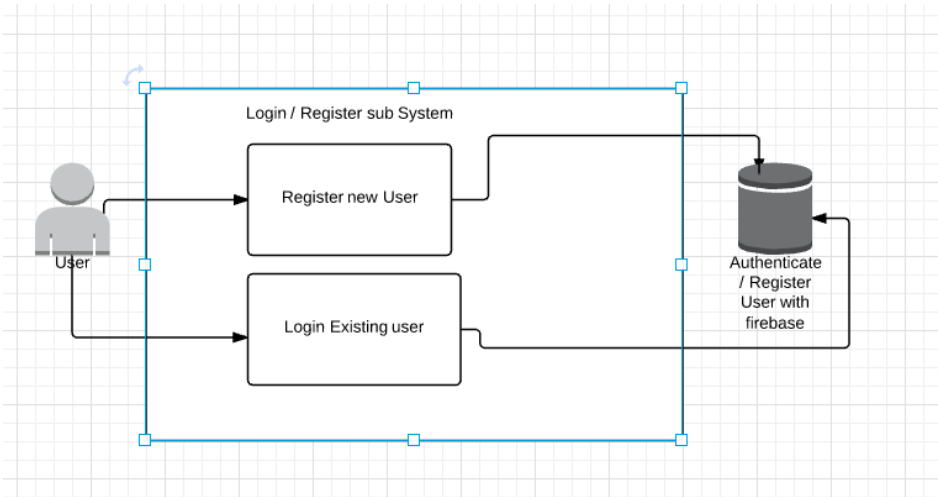
User interaction with Android application

##### Actors

- 1. User
- 2. Database

##### Use Case Diagram

\*diagram shows log in and register use case



##### Flow Description

### **Precondition**

The application must be installed and a user registered.

### **Activation**

Active upon launching of application by a registered user.

### **Main flow**

1. User Launches app
2. User enters login details
3. The system verifies these details and welcome screen is launched

### **Alternate flow**

A1 : Unregistered user

1. The system detects an unregistered user launches the app
2. The user is prompted to sign in or register
3. User completes registration or signs in
4. The use case continues at position 1 of the main flow

### **Exceptional flow**

The user exits the app before they have clicked register bike.

### **Termination**

Application closed by user.

### **Post condition**

The system goes into a wait state until user interaction is detected.

```
LoginUser(String email, String password){
    FirebaseAPI.LoginNewUser(email,password){
    {
        If(credentialsRecognised)
            SignInUser()
        else
            DisplayAppropriateFailureMessage()
    }
    }
}
```

```
}  
}
```

**Login user sudo code from SignIn.java class**

## **2.1.5 Requirement : Register a bike**

### ***2.1.5.1 Description & Priority***

Once the application has been downloaded and installed on a user smartphone the first screen a user will be presented with is a register bike screen. This will be a page which allows a user to enter all identifying bike details along with upload a photo of said bike. Upon pressing complete the bike data will be stored in our online DB. This is a high priority feature.

### ***2.1.5.2 Use Case 3***

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

#### **Scope**

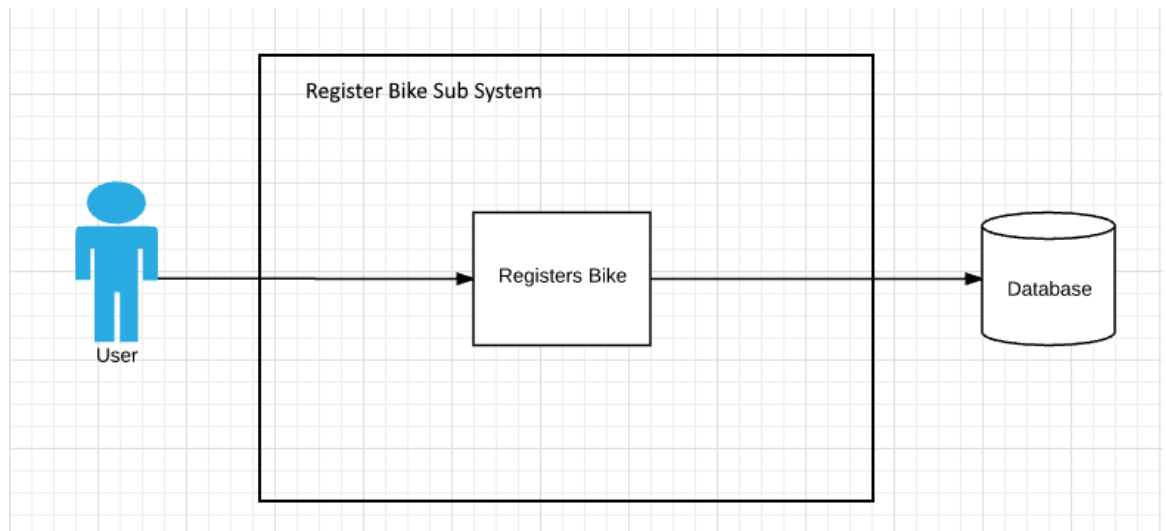
User interaction with Android application

#### **Actors**

1. User
2. Database

#### **Use Case Diagram**





### Flow Description

#### Precondition

The application must be installed and a user registered.

#### Activation

Active upon launching of application by a registered user.

#### Main flow

1. User Launches app
2. User enters bike details
3. The system saves these details to Firebase Database.

#### Alternate flow

A1 : Unregistered user

1. The system detects an unregistered user launches the app
2. The user is prompted to sign in or register
3. User completes registration or signs in
4. The use case continues at position 1 of the main flow

#### Exceptional flow

The user exits the app before they have clicked register bike.

#### Termination

Application closed by user.

### **Post condition**

The system goes into a wait state until user interaction is detected.

```
RegisterBike (myBike){  
    //all bike details gotten from user input  
    myBike;  
    //create DB instance and push bike to remote DB  
    FirebaseDatabase db= getInstance.getReference;  
    db.push().setValue(myBike)  
}
```

**Register bike sudo code**

## **2.1.6 Requirement : Edit a Bike / Register as Stolen**

### ***2.1.6.1 Description & Priority***

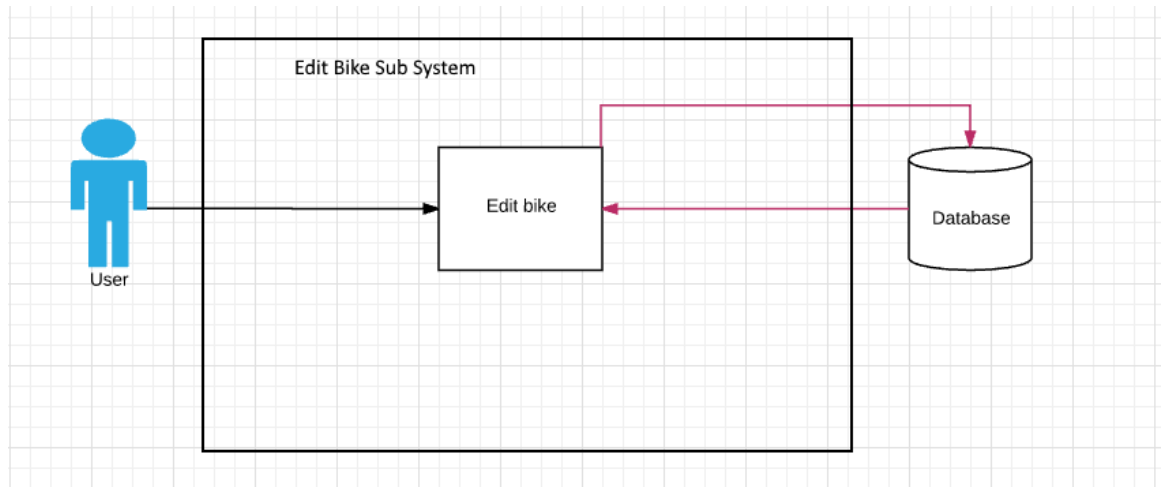
On this screen of the application a user may select one of the bikes they have previously registered and then edit all attributes of this bike. A user may also mark this bike as stolen. This is a high priority feature.

### ***2.1.6.2 Use Case 4***

#### **Scope**

User interaction with Android application.

## Use Case Diagram



### Flow Description

#### Precondition

The application must be installed and a user registered.

#### Activation

Active upon launching of application by a registered user.

#### Main flow

1. User Launches app
2. User navigates to edit bike screen
3. User selects appropriate bike
4. User updates chosen bike attributes including if stolen if applicable.
5. The system saves details, DB is updated.
6. If bike is marked as stolen, bike data is stored in new node in our JSON DB specifically for stolen bikes

#### Alternate flow

##### A1: Unregistered user

1. The system detects an unregistered user launches the app
2. The user is prompted to sign in or register
3. User completes registration or signs in

4. The use case continues at position 1 of the main flow

#### A2: No Bikes Registered

1. Start at position 2 of main flow
2. List of possible bikes is empty as user has never registered any bikes.
3. User needs to register a bike before this feature can be used.

#### Termination

Application closed by user.

#### Post condition

The system goes into a wait state until user interaction is detected.

```
EditBike (myBike){  
  
    //create DB instance and retrieve bike from remote DB  
    FirebaseDatabase db= getInstance.getReference;  
    myBike = db.getValue(myBike)  
  
    //populate UI fields with retrieved bike data  
    SetUI_Fields(myBike)  
    If(stolen)  
        Add to DB under root node stolen  
}
```

**Edit Bike or list as stolen sudo code**

## 2.1.7 Requirement : Viewing Stolen Bike Database.

### 2.1.7.1 Description & Priority

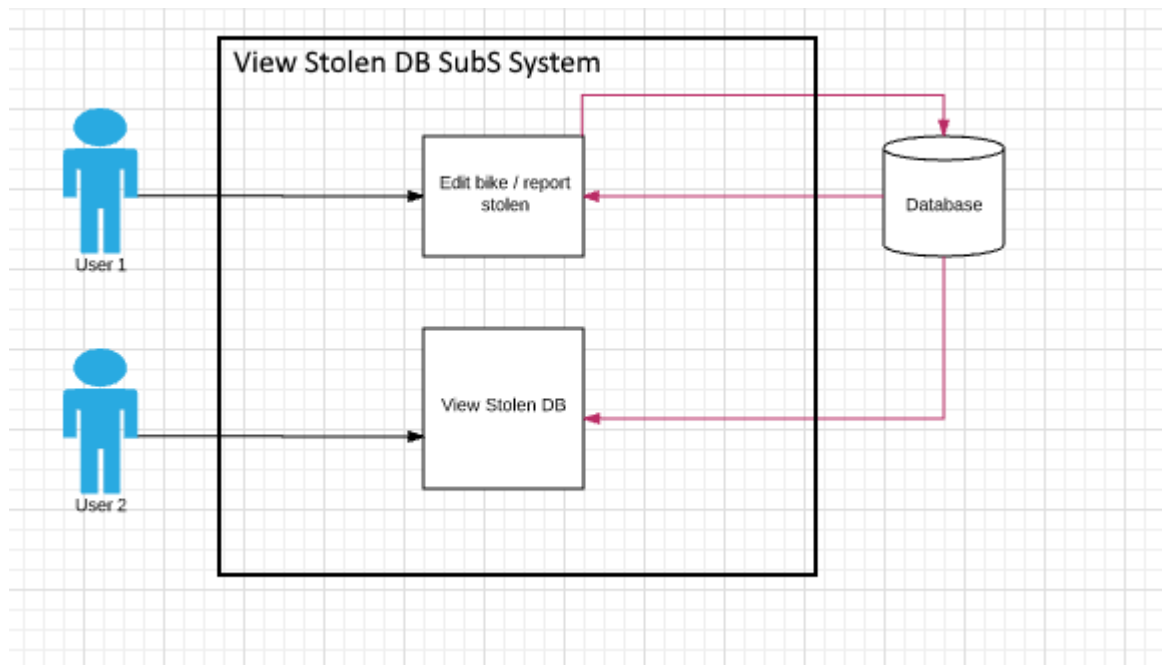
On this screen of the application a user may View all Bikes that if been reported stolen by any user of this application. AS this list could become quite large over time it is important to allow a user to query by data such as area code

### 2.1.7.2 Use Case 5

#### Scope

User interaction with Android application.

#### Use Case Diagram



#### Flow Description

#### Precondition

The application must be installed and a user registered.

## Activation

Active upon launching of application by a registered user.

## Main flow

1. User Launches app
2. User navigates to stolen bike DB screen
3. User Is presented with a list of bikes that have been listed as stolen

## Alternate flow

A2: No Bikes listed as stolen

1. User navigates to Stolen Bike DB screen
2. List of possible bikes is empty as no stolen bike data is present in DB
3. A user must first register a bike and then have it listed as stolen for this feature to be populated with data

## Termination

Application closed by user.

## Post condition

The system goes into a wait state until user interaction is detected.

```
RetrieveAllStolenBikes (myBike){  
  
    //create DB reference to stolenDB  
    FirebaseDatabase db= getInstance.getReference.stolenDB;  
    ListView myList;  
  
    //create list view Adapter  
    ListView Adapter(db){  
        setUI(db)
```

```
}  
  
myList.setAdapter(Adapter)  
}
```

**View stolen DB stolen sudo code**

## **2.1.8 Requirement : Reporting a sighting.**

### ***2.1.8.1 Description & Priority***

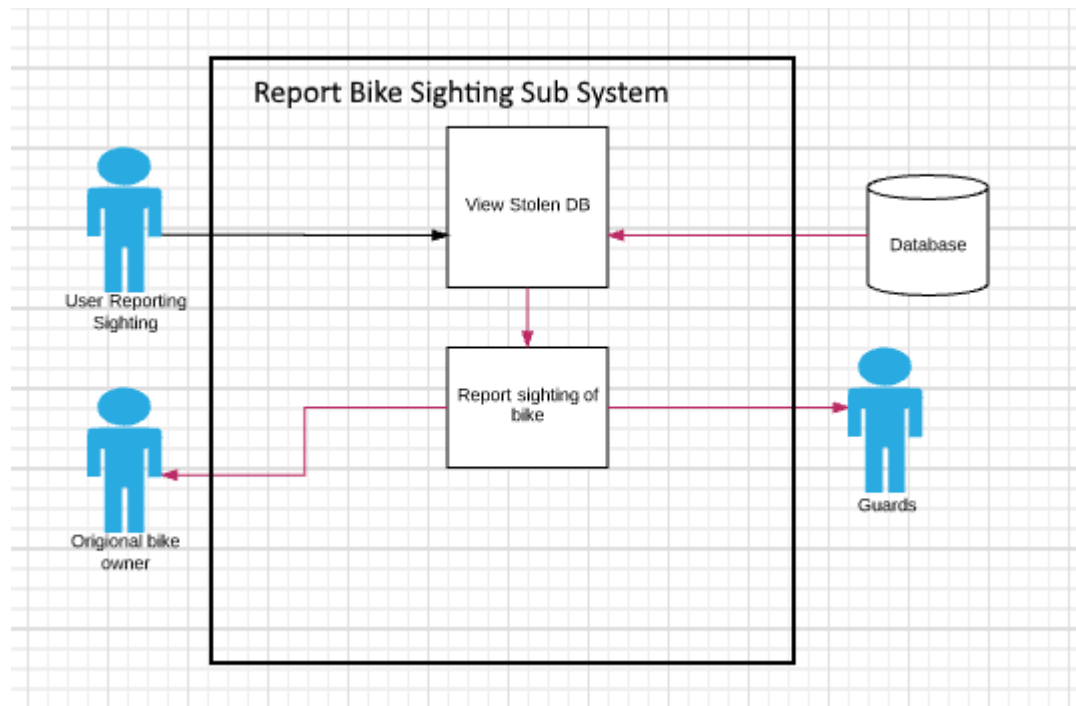
When viewing the stolen bike DB a user may report if they believe they have seen a bike that has been previously reported stolen this will open there email client and a report will be generated that can be sent to the original bike owner and a local guard station.

### ***2.1.8.2 Use Case 6***

#### **Scope**

User interaction with Android application, Email client, Interaction with another User who first reported bike stolen,

## Use Case Diagram



### Flow Description

#### Precondition

The application must be installed and a user registered.

#### Activation

Active upon launching of application by a registered user, navigating to Stolen Bike DB screen and selecting one of the bikes.

#### Main flow

1. User Launches app
2. User navigates to stolen bike DB screen



3. User Is presented with a list of bikes that have been listed as stolen
4. User selects one of the bikes indicating they have seen this bike.
5. Users Email client is launched with email body pre generated with all bike details and guards and original user who reported it stolen as recipients.

### **Exceptional flow**

The user has no email capable application on their smartphone

### **Termination**

Application closed by user.

### **Post condition**

The system goes into a wait state until user interaction is detected.

```
//retrieve list of all bikes stolen first same as above step
RetrieveAllStolenBikes (myBike){

    //create DB reference to stolenDB
    FirebaseDatabase db= getInstance.getReference.stolenDB;

    ListView myList;

    //create list view Adapter
    ListView Adapter(db){
        setUI(db)
    }

    myList.setAdapter(Adapter)
}
```

```
//Report specific bike stolen if clicked on  
Mylist.setOnClickListener(){  
    Prompt user to confirm  
    If(confirm)  
        notifyOriginalUserInApp()  
}
```

**Report bike stolen sudo code**

## **2.1.9 Requirement : Google Maps Integration.**

### ***2.1.9.1 Description & Priority***

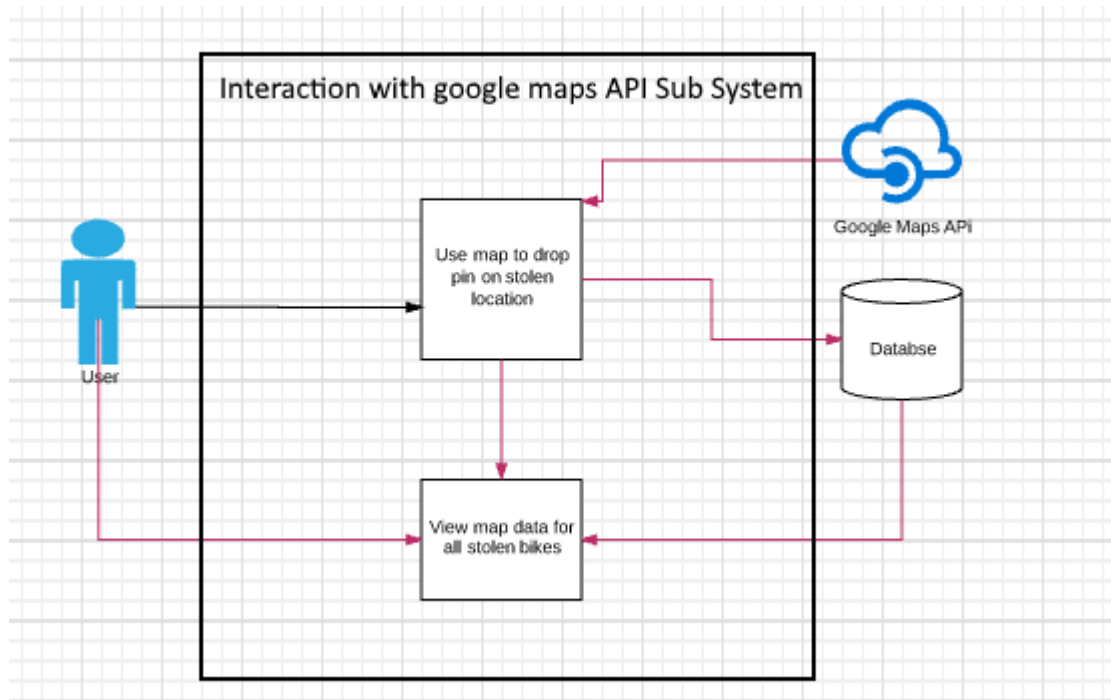
When a user edits a bike, If they select a bike as stolen they then have the option of using google maps to drop a marker on the area where the bike was stolen. This data is then saved in the DB and is available for any user to see I n a map view. Overtime this will build Blackspot areas that will enable users to avoid certain areas.

### ***2.1.9.2 Use Case 7***

#### **Scope**

User interaction with Android application, Google maps API.

#### **Use Case Diagram**



## Flow Description

### Precondition

The application must be installed and a user registered.

### Activation

Active upon launching of application by a registered user, navigating to Edit bike screen, selecting a bike as stolen, then selecting upload map data.

### Main flow

1. User Launches app
2. User navigates to edit bike screen
3. User selects appropriate bike
4. User Selects that bike has been stolen.
5. User selects to input Map data.
6. Screen launched that allows user to drop a marker on map overlay.
7. Marker data saved and uploaded to DB.

## Termination

Application closed by user.

## Post condition

The system goes into a wait state until user interaction is detected.

```
// when Editing a bike.

EditBike (myBike){

    //populate UI fields with retrieved bike data

    Set_UI_Fields(myBike);

    If(stolen)

        Add to DB under root node stolen

        Set last seen address

        Geo coordinate API gets latitude and longitude of address

    }

//on maps screen

populateMap{

    get instance of google maps

    get latitude and longitude of all stolen bikrs

    use this data to populate marks on map and create hot spots.

}

Google maps Integration sudo code
```

## 2.1.10 Requirement: Bluetooth Sensor Integration.

### 2.1.10.1 Description & Priority

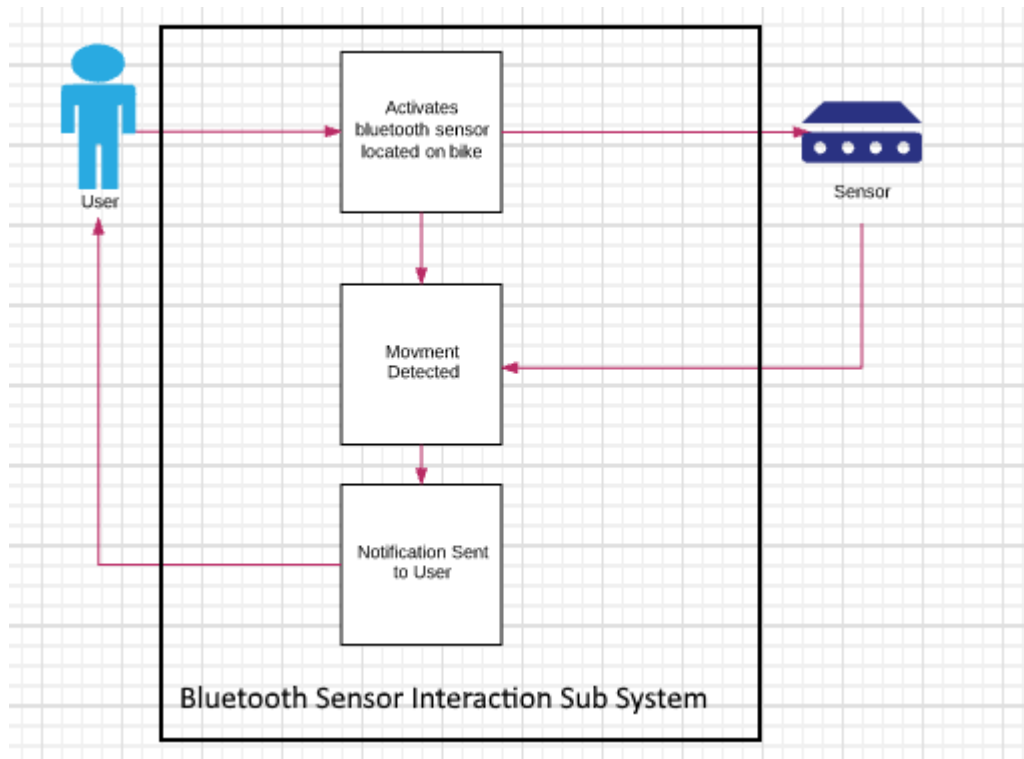
Users have the option of attaching a Bluetooth sensor to their bike which will alert them of unauthorized movement of bike. This is designed to be used when a bike is located a short distance away from you such as outside work or college and you will be notified via smartphone that your device is being moved without your permission.

### 2.1.10.2 Use Case 8

#### Scope

User interaction with Android application, Bluetooth sensor.

#### Use Case Diagram



## **Flow Description**

### **Precondition**

The application must be installed and a user registered. User must have sensor placed on their bike.

### **Activation**

Before leaving their bike in an area they will be close to (College, home) A user indicates in app that bike is “parked” they will then be notified if bike is moved outside sensor range.

### **Main flow**

1. User Launches app
2. User selects activate sensor
3. Device recognizes that the sensor is within range of phone
4. Phone will be notified if bike is moved

### **Termination**

Application closed by user. Phone powered off. Unexpected hardware failure of Bluetooth sensor.

### **Post condition**

The system goes into a wait state until user interaction or Bluetooth sensor interaction is detected.

```
// link beacibs  
  
linkSensor(){  
  
    establish connection to beacon
```

```
If (link is broken)

    alert user (send notification and make UI change)

else(connection established)

    alert user (send notification and make UI change)

}

Link sensor sudo code
```

## **2.1.11 Requirement: Scanning for stolen bikes using Bluetooth sensors and users device.**

### **2.1.11.1 Description & Priority**

Users may use their mobile device to scan the surrounding area for any bikes that have been registered as stolen. The user's device scans for a unique ID that the sensor emits if this matches that of a bike listed as stolen the UI is updated.

If bikes registered as stolen and using the sensors are in the area the user will be able to see the details of that including description and its last known location.

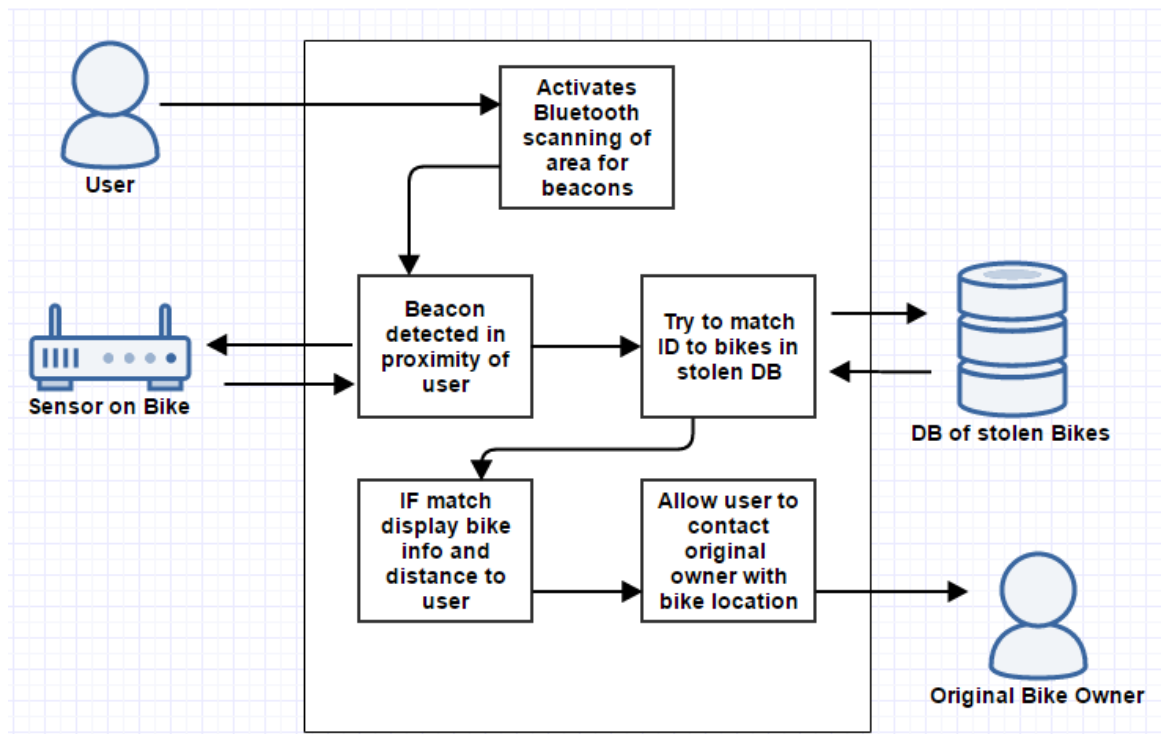
This feature will be able to provide an approximate distance to the location of the bike which will update live as a user moves closer to the sensor which would be hidden in the frame of the bike. This will allow a user through trial and error to successfully pin point the exact physical location of a bike listed as stolen

### 2.1.11.2 Use Case 7

#### Scope

User interaction with Android application, Bluetooth sensor, other bikes and their owners.

#### Use Case Diagram



#### Flow Description

#### Precondition

The application must be installed and a user registered. User must have sensor placed on their bike.

#### Activation

User Navigates to “Link to bike” through the navigation menu and selects “search for stolen bikes”



## Main flow

1. User Launches app
2. User selects Sensor manager
3. User selects Scan for Stolen bikes
4. Device starts scanning the area for the Bluetooth frequency the beacons emit.
5. Sensor is detected, the phone reads the unique ID of the identified beacon and searches the stolen database for a match
6. If match is successful, All details of that bike are displayed to user along with last known location.
7. Signal strength of sensor is used to determine an approximate distance. This is displayed to the user.
8. As the user moves in the physical world distance is updated
9. User successful identifies the stolen bike
10. User send an email to original owner notifying them of the confirmed sighting of their bike. A user may also choose to ring the authorities at this point.

## Termination

Application closed by user. Phone powered off. Unexpected hardware failure of Bluetooth sensor.

## Post condition

The system will continue to scan the area until a user exits out of this screen.

```
ScanArea( List of nearby beacons){  
    For ( all beacons found in area ){  
        Compare any found beacons to bikes labeled as stolen in Database;  
        IF (beacon ID matches stolen bike){
```

```

        return all bike details and display to user;

        Use signal strength to calculate approximate distance of each bike
returned

        }//end if

    }//end for

} //end method

```

Scan area sensor sudo code

## 2.2 Data requirements

### 2.2.1 Database Design

For this project I am using googles Firebase platform as a backend solution. Data such as the DB of stolen bikes will be built over time as more users use application. The Data is all user generated. For the purpose of testing and demonstration I shall be providing a sample set of data.

#### Database details

<b>Database Provider:</b>	Google Firebase
<b>Name of database:</b>	Find_My_Bike
<b>location of database:</b>	<a href="https://findmybike-1a1af.firebaseio.com/">https://findmybike-1a1af.firebaseio.com/</a>
<b>Database Style:</b>	NoSQL cloud database
<b>Data Format</b>	JSON

Firebase is a JSON database, meaning all your data is stored as JSON objects. There are no tables or records as in a traditional SQL database. Only a JSON tree. Java objects can be passed to this tree where they become a node on the tree structure. Nodes can be added as children of other nodes for more complex storage.

The main data structure stored in my database is a user's bike. Based on if a user has listed their bike as stolen or not, Stolen bikes are moved to a different node on the JSON database. I also store user profile information. What follow are some screenshots of the database and particular data structures used.

## Database high level overview

findmybike-1a1af

[-] Bikes Registered By User

[+] ronanbrowne88

[-] testing

[+] -KUb1IGNVYUBE2UzfPKP

[+] -KUb4h4j03IHwb8MnTQP

[+] -KUb06h1QOPBqlj0mq8g

[-] -KUb08LVbV8p2BfunaeB

color: "Black "

frameSize: 29

imageBase64: "iVBORw0KGgoAAAANSUHEUgAAARoAAACzCAIAAADZtUR/AA#

lastSeen: "Wellpark, galwa

latitude: 53.283114

longditude: -9.029377

make: "Gary Fisher

model: "G200"

other: "Racer handlebars

otherFeatures: "Racer handlebars

stolen: true

[+] -KUbb0msoSL20LU9-lkq

[+] -KXC0gkYtAC1SjJ4gvbz

[+] -KXKsyhluXbaeVimapjB

[+] -KXQ9L4b3qltnluSpmaP

[+] -KXQArCpXKoR2L9v9Jsd

[+] x13114522

[+] QueryResults

[+] Stolen Bikes

[+] User Profile Data

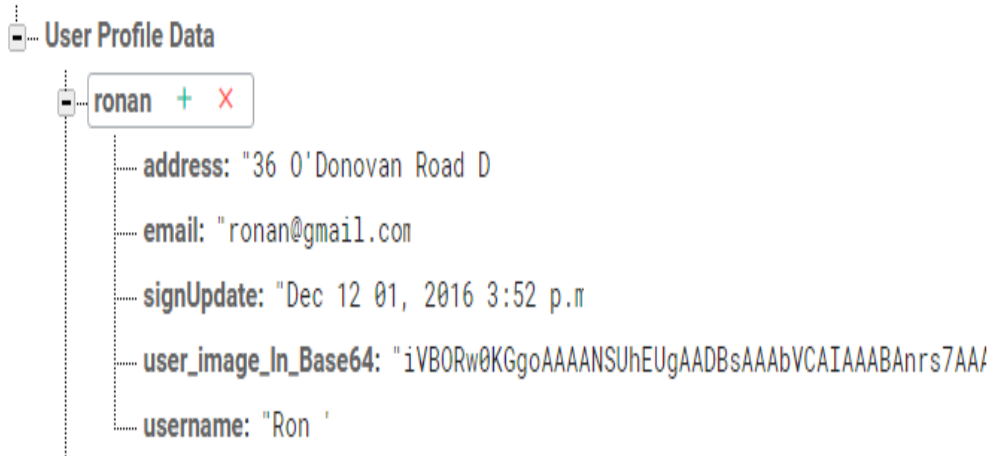
## Bike Data Structure

The root node here is a randomly generated key for ID purposes. The child attributes are all the values that may be associated with a particular bike. A bike in the DB maps directly to a Bike object in the Java code.

```
-KUb06h1QOPBqlj0mq8g
  color: "Purple"
  frameSize: 6
  imageBase64: "iVBORw0KGgoAAAANSUhEUgAAAOEAAADhCAIAAACx0UUtAA/"
  lastSeen: "Cork"
  latitude: 51.896891
  longitude: -8.486315
  make: "Custom Made"
  model: "32a"
  other: "Rear seat"
  otherFeatures: "Rear seat"
  registeredBy: "testing"
  stolen: true
```

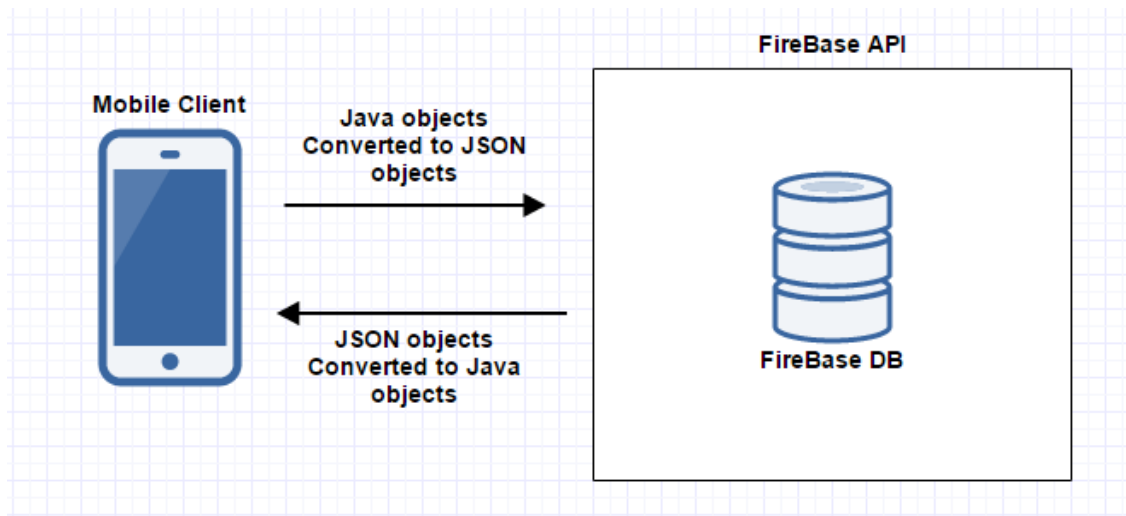
## User Data Structure

Here we have an example of a user data structure in the DB. This maps to a user object in the java code.



## 2.2.2 Data Conversions

The System must convert data from Java to JSON and from JSON to Java. This is necessary as the database used is a NoSQL JSON database so java objects must be mapped to JSON before being persisted and the JSON must be converted to Java when returned to system. This is handled by the Firebase API. Image 4.3.1 below depicts conversion of data.



## **Image 4.3.1 – Data Conversion**

### **2.3 User requirements**

Focus groups with users lead to identifying 3 key areas which the majority of users requested these were

Cross platform availability: Users wished for the app to be available on Apple and Android mobile devices. However, the current iteration of the project will focus on the android platform due to time and budgetary constraints. Future Versions of the project will expand to include other mobile platforms.

Usability: Users requested the app to be intuitive. Navigation between each feature should be obvious. The application should have a gentle learning curve, allowing for use by advanced smartphone users or novice users.

Security: users also expressed concerns about the security of their data and how it will be stored. These concerns will be addressed by Googles Firebase the encrypted secure database handles security aspects of storing data safely.

### **2.4 Environmental requirements**

#### Internet:

The application will require an active internet connection to get the latest data from cloud storage, Features may still be used without an internet connection but a user must be made aware that they may not be seeing the most up-to-date data available.

### Bluetooth:

If a user wishes to listen for any unauthorized movement of their bike using the external sensors this will require an active internet connection. A user may still use the other features of the application if they not wish to turn on Bluetooth.

## **2.5 Usability requirements**

Mobile users are very goal orientated, they expect instant results from applications, delays and lags are unacceptable for a mobile user, even more so than a traditional desktop user. If an application lags or is not easy to use the user will look for a competitor's application that may perform better.

### Navigation:

Interfaces and menus should be easy to understand. Navigation system should be intuitive and consistent. The user should be able to return to the home page of the application easily from any screen. For this project this will be achieved through a navigation drawer which can be accessed from any screen of the application.

The Navigation area should have no scrolling features. The user should not have to scroll horizontally or vertically to see all available menu options.

Access to the menu system should be kept in a prime location on the system, in this case we will keep the button to open our menu in the top left of the system. This is keeping in line with Googles Material Design recommendations.

### Learning:

The system should be designed with the maximum learnability in mind. Although all features will be designed with simplicity in mind. Detailed documentation will be provided upon release of system for users to reference. This documentation should be kept up to date as features change or new features are added.

### Error handling:

If an error should occur during the running of the applications an error message will explain the steps needed to recover from this error and the likely cause if it is identifiable.

### Confirmation from user:

Permanent actions such as deleting a bike should ask for confirmation from the user to ensure the correct action is taken.

### Colour:

The colour scheme of the application will be designed in a way which is appealing to the broadest demography of users. It was decided during the course of development that the app should allow users to select a theme which would completely change the colour scheme depending on the preference of the user.

### Non-intrusive:

The system shall not use any pop ups or large advertisements. These are widely disliked by users and use of such features will make the application less appealing to customers.

### Form Entry:

On any screen where a data is being gathered from a user via a form (bike registration, bike editing) the gathering of information should be as streamlined as possible.

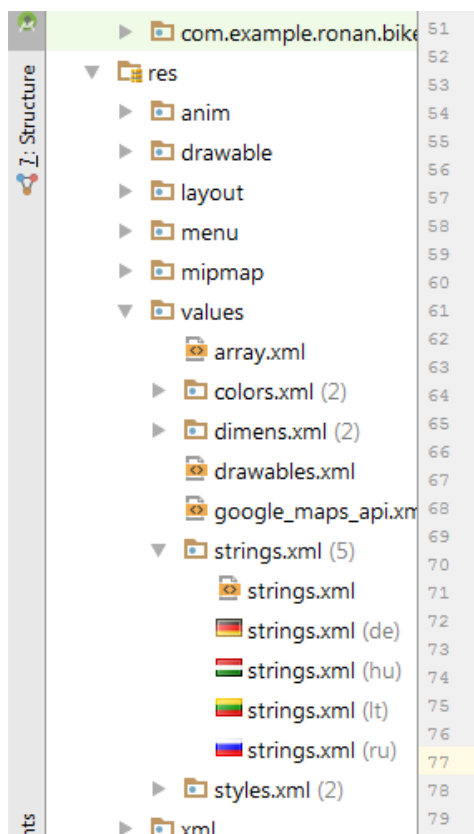
The simplest and most appropriate input method must be chosen for each field, for example a drop down menu would not be appropriate for a selecting a user's name however when selecting the frame size of a bike a drop down menu of sizes may be more appropriate.



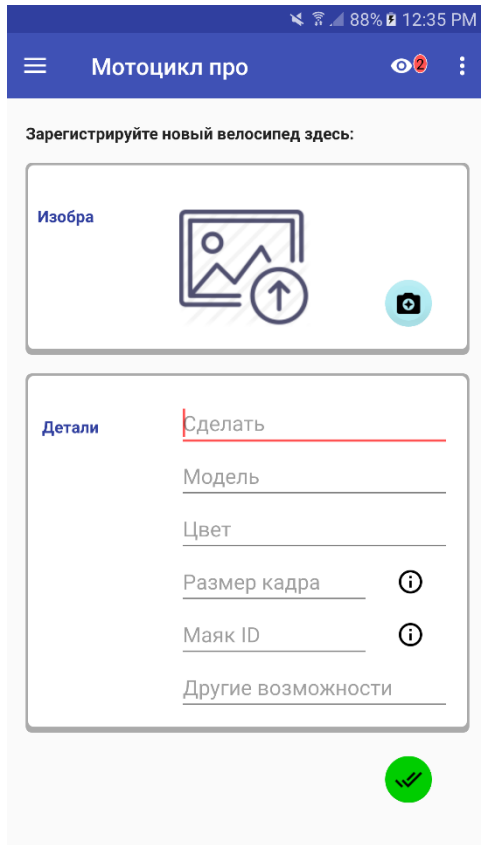
If a field requires some particular data from a user, such as an email address the system should validate that the user had entered the expected data. The system should not allow alphabetical characters to be entered on any field which requires only numerical data. Anywhere in the system where a user is selecting a date, the application should provide the user with a visual representation of a calendar for selecting dates.

### Internationalization

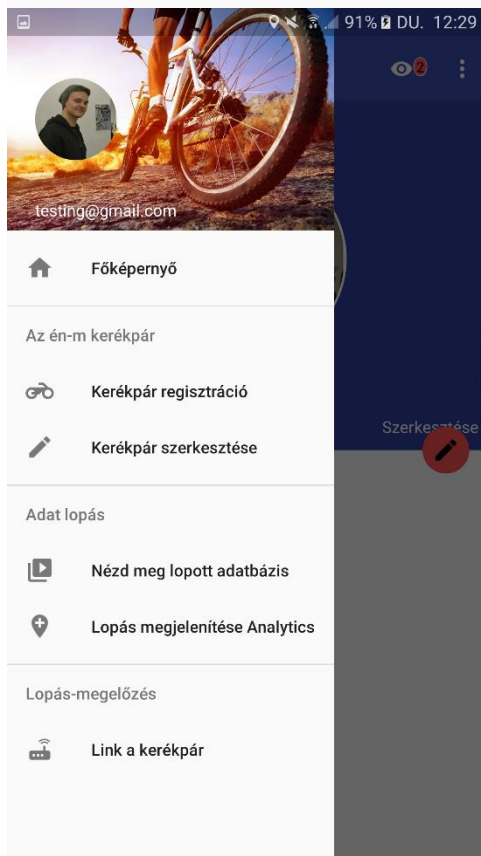
It is important that the application be available to more than just the English speaking community. Upon initial release the application will support English, German, Hungarian, Lithuanian and Russian. Below is a screenshot of the project structure showing the Strings.xml files for the various supported languages.



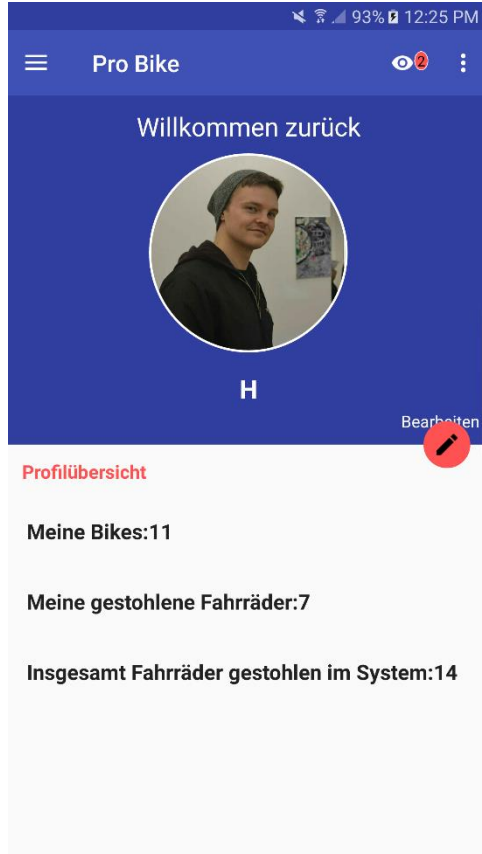
Project structure showing the String.xml file of the various other languages supported



Here we see a screenshot of the application running in Russian.



Here we see the application run Hungarian.



This screenshot shows the application running when German is selected as the default language.

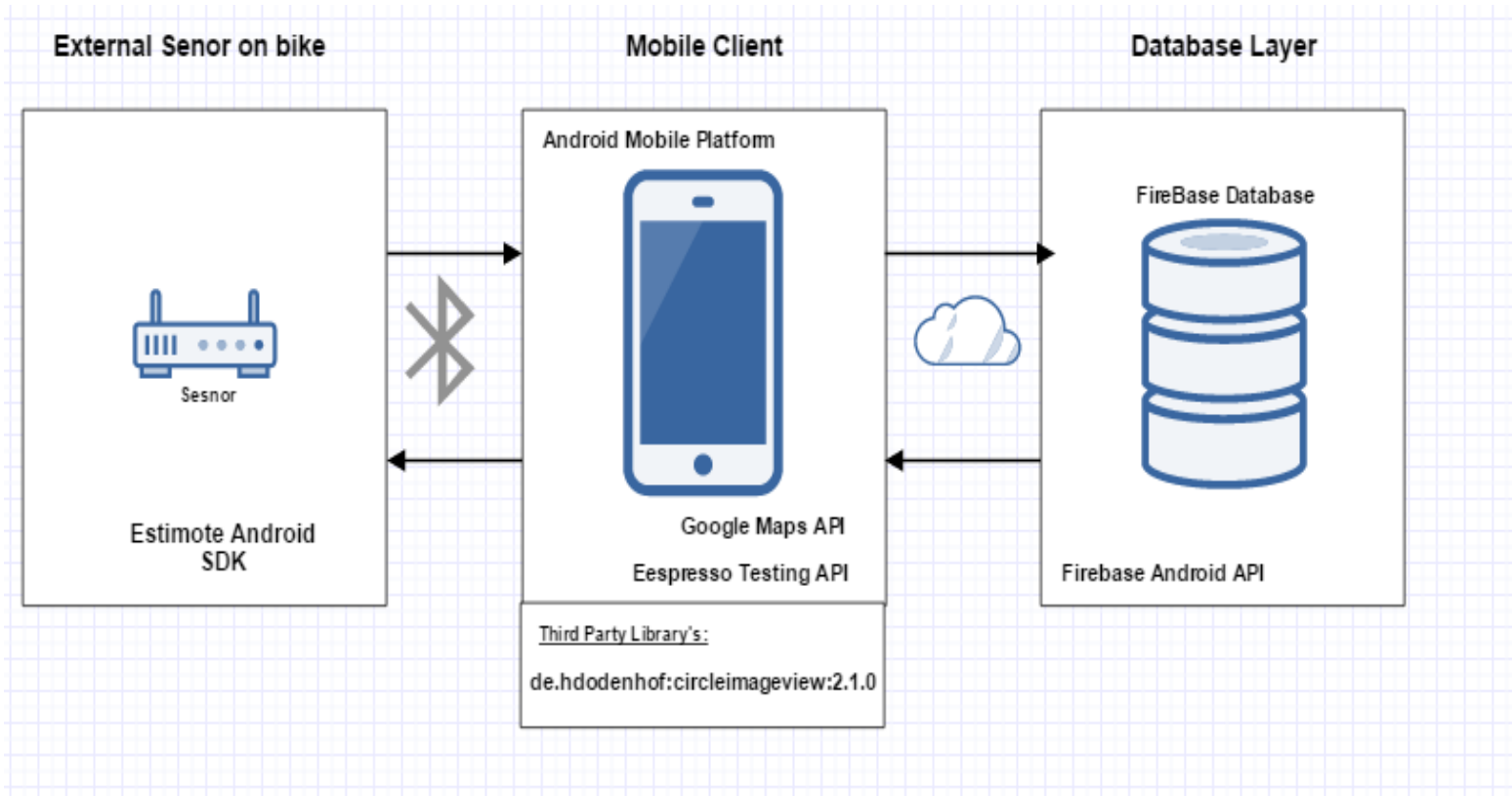
## ***2.6 Design and Architecture***

### **2.6.1 Architecture Design**

The application will consist of a Java-based mobile client which is developed with the Android platform which connects with a remote Firebase backend database.

A user's device will also directly communicate with Bluetooth sensors when using the anti-theft features of the application. What follows is an Architectural diagram of the system along with a brief explanation of each aspect of the system.

## Architecture Diagram:



### **Firestore DB**

Cloud hosted NoSQL JSON database.

### **Firestore API**

This is how our mobile application will communicate with the firestore database, it allows for the mapping of java objects to JSON objects in the database.

### **Google maps API**

Will allow our application to use features such as geolocation addresses, placing markers and heat map data.

### **Espresso API**

This is a testing API which allows automated testing of android UI components

## **Android Mobile Platform**

This will be the application running on the user's android device

### **Third party library's:**

The GIT repo Circle Image View (<https://github.com/hdodenhof/CircleImageView>) is used in the application this provides classes which allow for a circular image property as opposed to the default rectangular image

### **Estimote SKD**

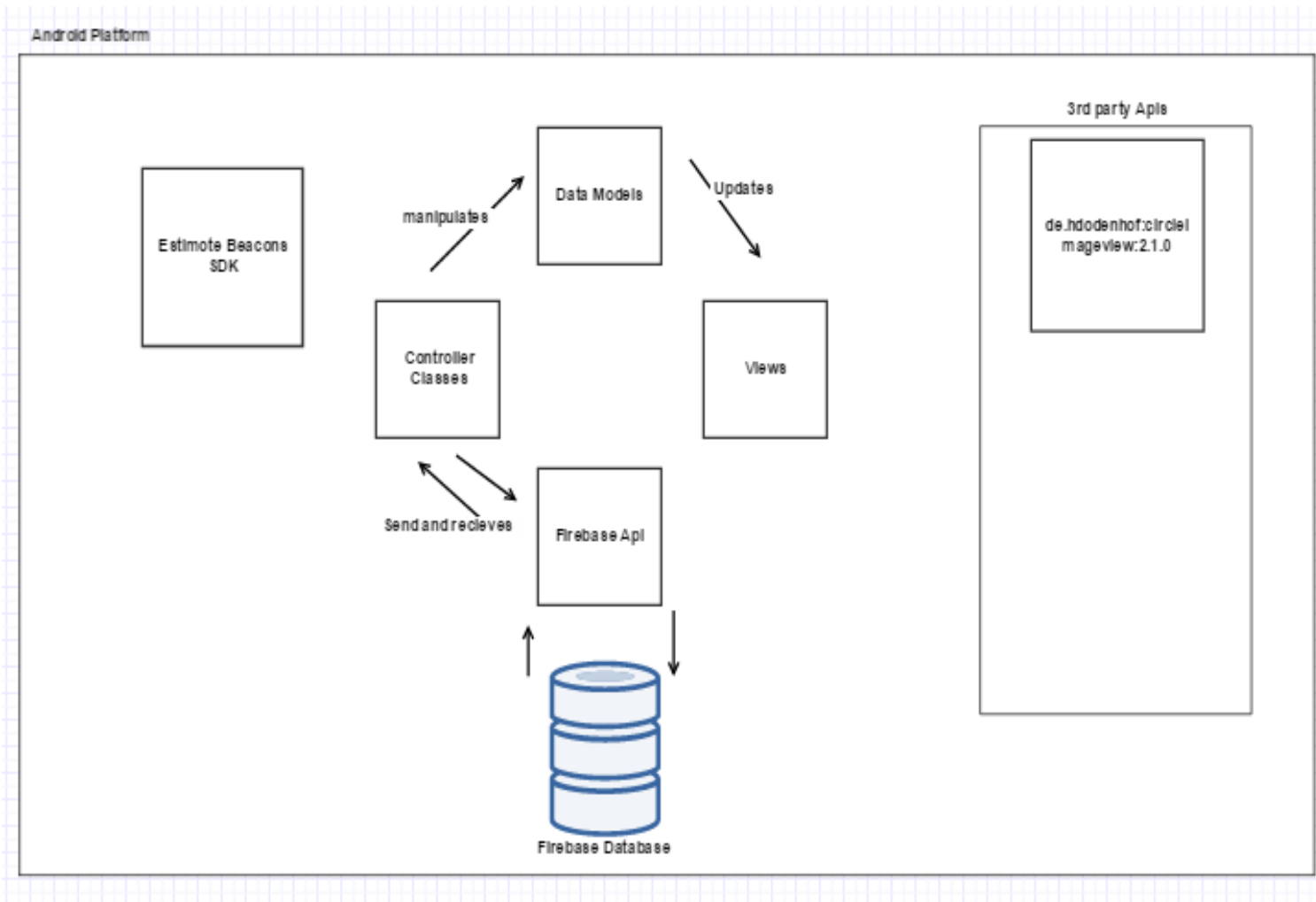
This is the SDK that provides tools for interaction between the sensors and a user's phone. The SDK allows us to use classes such as BeaconManager.java which can pick up nearby sensors.

## **2.6.2 Software Architecture**

Classes and packages in the system should aim to have loose coupling (a measure of dependence between classes or modules). Classes do not need to exist in isolation but should not be heavily dependent on each other and communication between modules or classes should happen through well-defined interfaces.

The software should have high cohesion, software modules (classes, libraries) should have responsibilities that are strongly related. This will allow for maintenance and reusability of the system to be achieved more easily as dependencies are minimized.

Image 3.3.1 below lays out the software architecture of the application. What follows is a brief explanation of each part.



**Image 3.3.1 - Diagram of software architecture of project**

Android Platform: Google's open and free software stack that includes an operating system, middleware and also key applications for use on mobile devices.

Data Models: These are java classes which represent the data used in the application. In the case of this application they are the user's data and bike data. These data models are then mapped to the Firebase remote DB when a user calls the write functionality. These data models are laid on in the database architecture section of this document (See section 4.2)

Views: Refer to the physical components the user interacts with. The UI widgets such as text views and buttons. In Android the views are set out in XML files. This allows us to separate UI creation and data manipulation allowing for cleaner code.

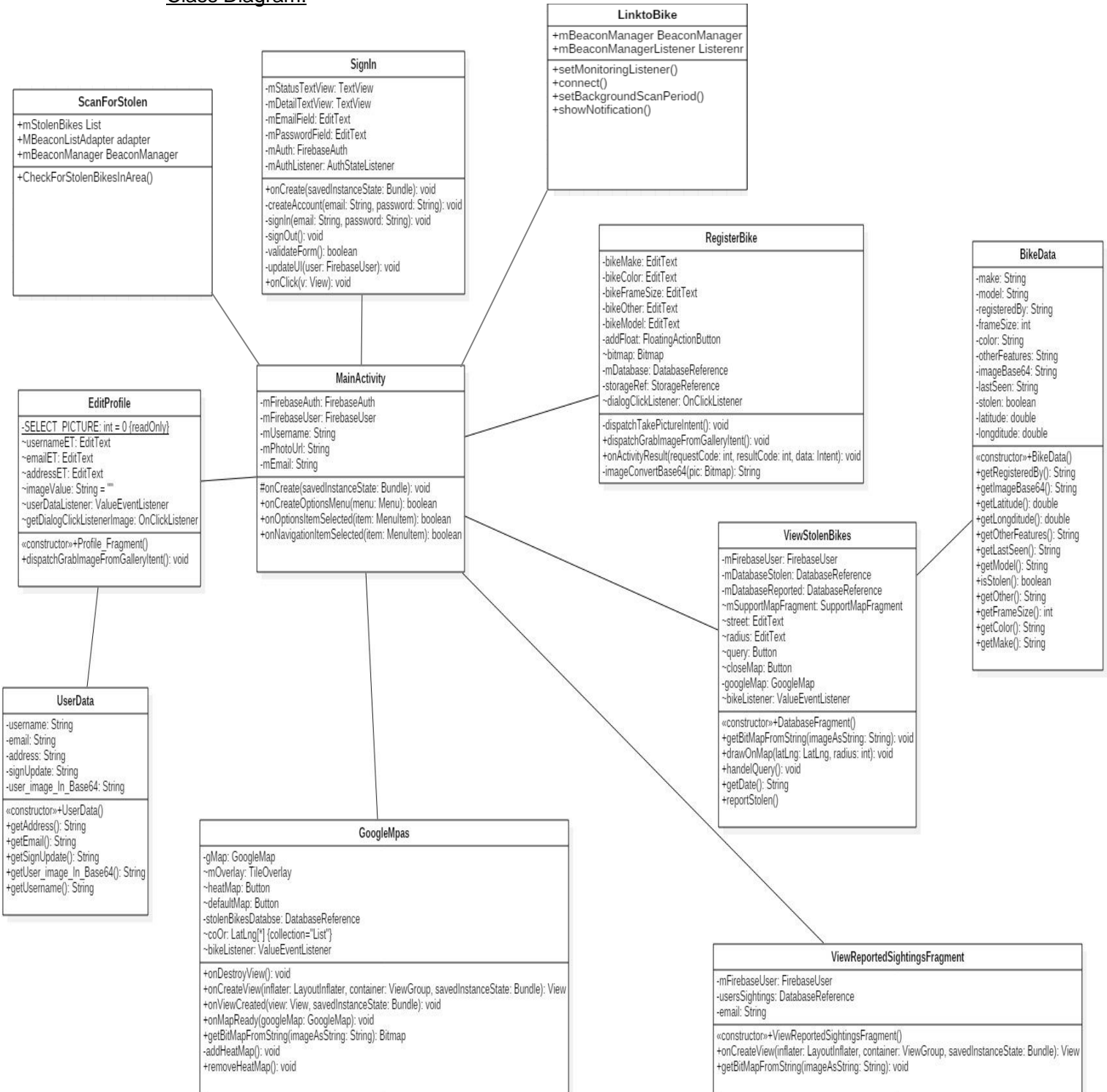
Controller: The Controller classes populate the view with data from the model. The controller also monitors and interprets actions such as button clicks and then takes appropriate action.

Firebase API: This is how our mobile application will communicate with the firebase database, it allows for the mapping of java objects to JSON objects in the database.

Estimote SDK: This is the SDK that provides tools for interaction between the sensors and a user's phone. The SDK allows us to use classes such as BeaconManager.java which can pick up nearby sensors.

Circle Image View: Third party library used to modify the default android image view widget to be circular.

## Class Diagram:





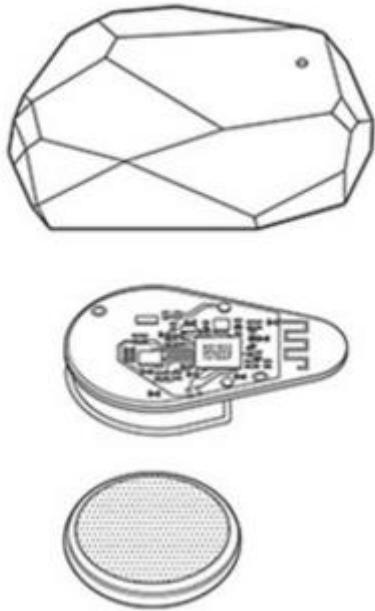
### 2.6.3 Hardware Architecture

This project shall make use of external sensors known as beacons. These beacons are provided by hardware manufacturer Estimote see image 3.1.

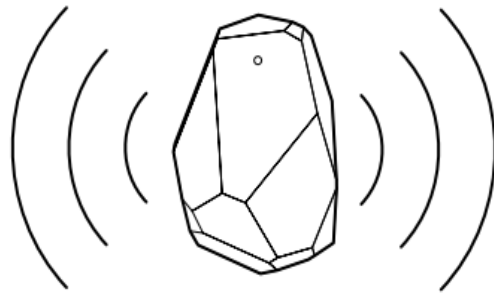
#### Beacon Specifications

- Micro-computer powered by an ARM chip.
- Motion and temperature scanners.
- NFC compatibility.
- Transmit signals through 2.4GHz frequency of Bluetooth
- Range of up to 70m.

#### ESTIMOTE BEACON



**Image 3.2.1 – Estimote beacon hardware breakdown**



Android 4.3+ devices

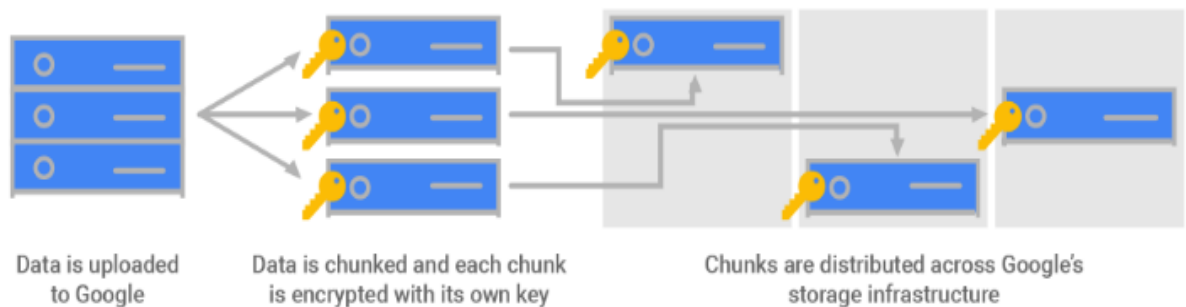
### **Image 3.2.2 – Beacon Data Transition**

Image 3.2 above depicts a beacon transmitting a signal which is then picked up by a nearby android device. Each beacon has a Bluetooth low energy transmitter. It broadcasts radio signals over the air containing unique, location-specific data. The nearby Smart phone scans for these signals, if they are detected the system will carry out an appropriate action.

## 2.6.4 Security Architecture

Data stored on firebase uses googles encryption at rest default encryption standard. Google uses a method of dividing and distributing the data along with the Advanced Encryption Standard (AES) algorithm to encrypt data stored on its Firebase servers.

Image 3.4.1 below demonstrates how google separates and stores data on Firebase.



**Image 3.4.1** – source: <https://cloud.google.com/security/encryption-at-rest/default-encryption/>

### Authentication:

This application will require a user to be logged on to the system to make use of the applications key features. For Authentication the application will use the Firebase platforms built in authentication feature.

**How it works:** The user will create an account by providing a unique email address and password. The system will get these credentials from the users input. It is important to note these credentials are not stored locally.

They are then passed via the create user method in the SignIn.java class to the firebase API. Firebases cloud services will then create the new user account. Upon successful creation of an account a user is automatically signed in.

```
CreateUser(String email, String password){  
    FirebaseAPI.createNewUser(email,password){  
    {  
        If(Successful)  
            SignInUser()  
        Else  
            DisplayAppropriateFalureMessage()  
    }  
}
```

**Create new user sudo code from SignIn.java class**

Only once the user has been logged on to the system do they have read and write permissions for the database. Database rules shown below. The code shows once a user has been authorized, by their credentials being verified, only then are they allowed read and write permissions for data.

```
rules () {  
    if (authorization != null){  
        allowReadToDB()  
    }
```

```
allowWriteToDB()
}
}
```

Database rules for allowing reading and modifying of data.

### 2.6.5 Communication Architecture

The application will need to communication to external entities such as the Bluetooth sensors or cloud based database. Image 3.5.1 depicts the communication protocols used by system. What follows is a brief explanation of each one.

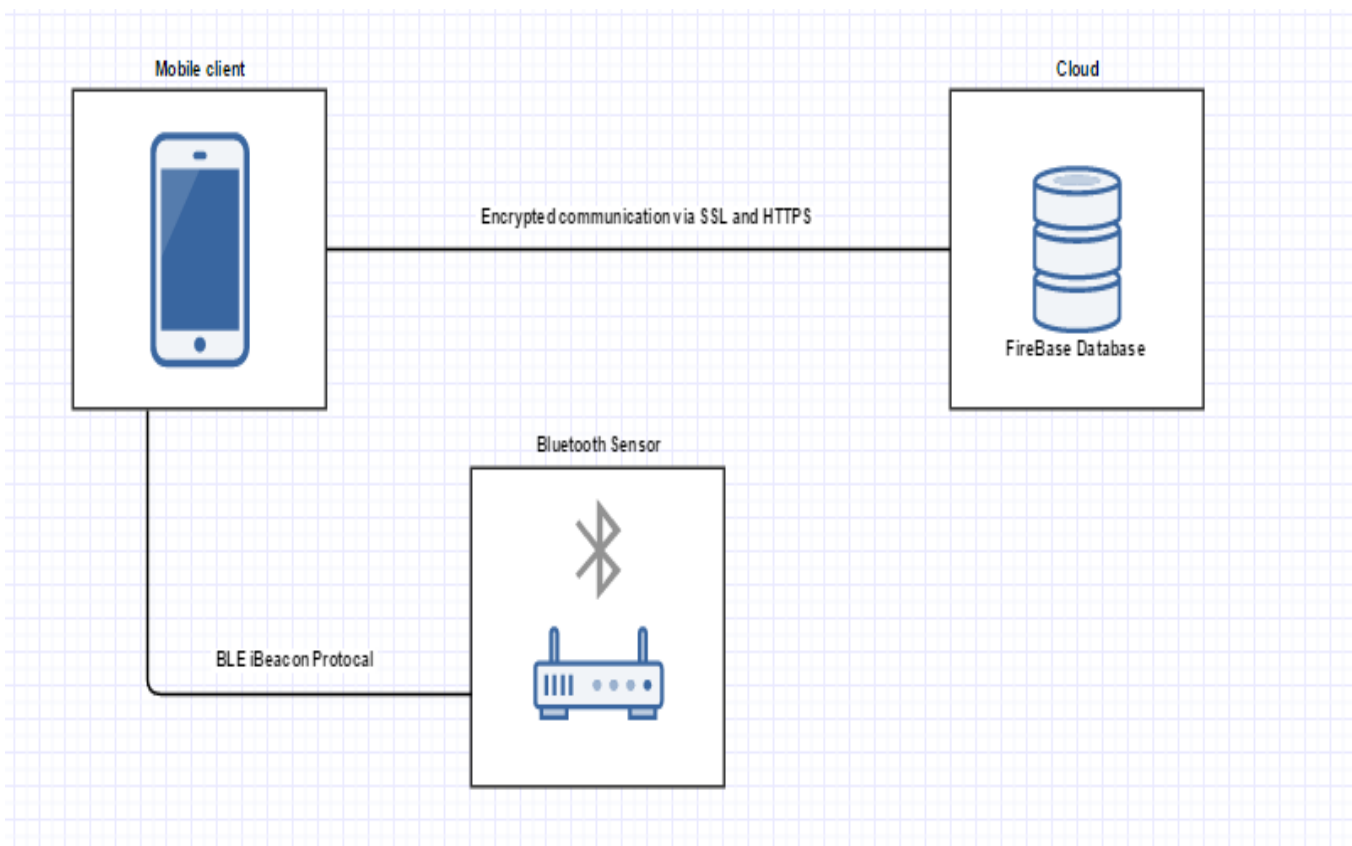


Image 3.5.1 – Communication architecture

Mobile client to Firebase database communication: Data is sent from the mobile client over the internet via https and SSL to the Firebase servers where it is stored in JSON format. The packets of data sent to and from Firebase are encrypted (see Security Architecture). The mobile clients

Mobile client to Bluetooth Sensors communication: Beacons use Bluetooth low energy (BLE) to transmit data, BLE uses standard communication protocols to allow devices to communicate. The sensors will use the iBeacon protocol to transmit structured packets of data that the users phone can listen for and perform actions depending on the presence or content of these packets.

The packets of data transmitted by the beacons contain the iBeacon ID, a 20 character unique key, and information about signal strength, a users phone can use this information to determine if a beacon is in range and determine how far away it is.

## 2.6.6 Application Program Interfaces

The following Application Program Interfaces (APIs) are used in the system.

Firestore API: The firestore API is provided by Google. It is added to the system, through a Gradle dependency (see Appendix B, Key terms). This API allows us to communicate with the remote firestore database and provides the functionality for mapping java objects to JSON and vice versa. This API also manages login / sign up functionality along with CRUD operations on data.

Google Maps API: This API allows us to use the Google maps platform in our system. It gives the user the ability to navigate, zoom and set markers on an interactive map.

Geolocation API: This API provides the functionality to convert a user inputted address into latitude and longitude co-ordinates. This is used in the system to provide the user with visual representations theft locations.

### 2.6.7 User Interface Design

When designing the user interface Googles Material Design Specification where kept in mind. Material design is Googles comprehensive guide for Visual and interactive design. Material design sets out interface standards which should be followed, these include spacing and positioning of UI elements. Image 4.5.1 below is a guideline image for implementing material design specifications, the red lines and numbers represent specific pixel spacing.

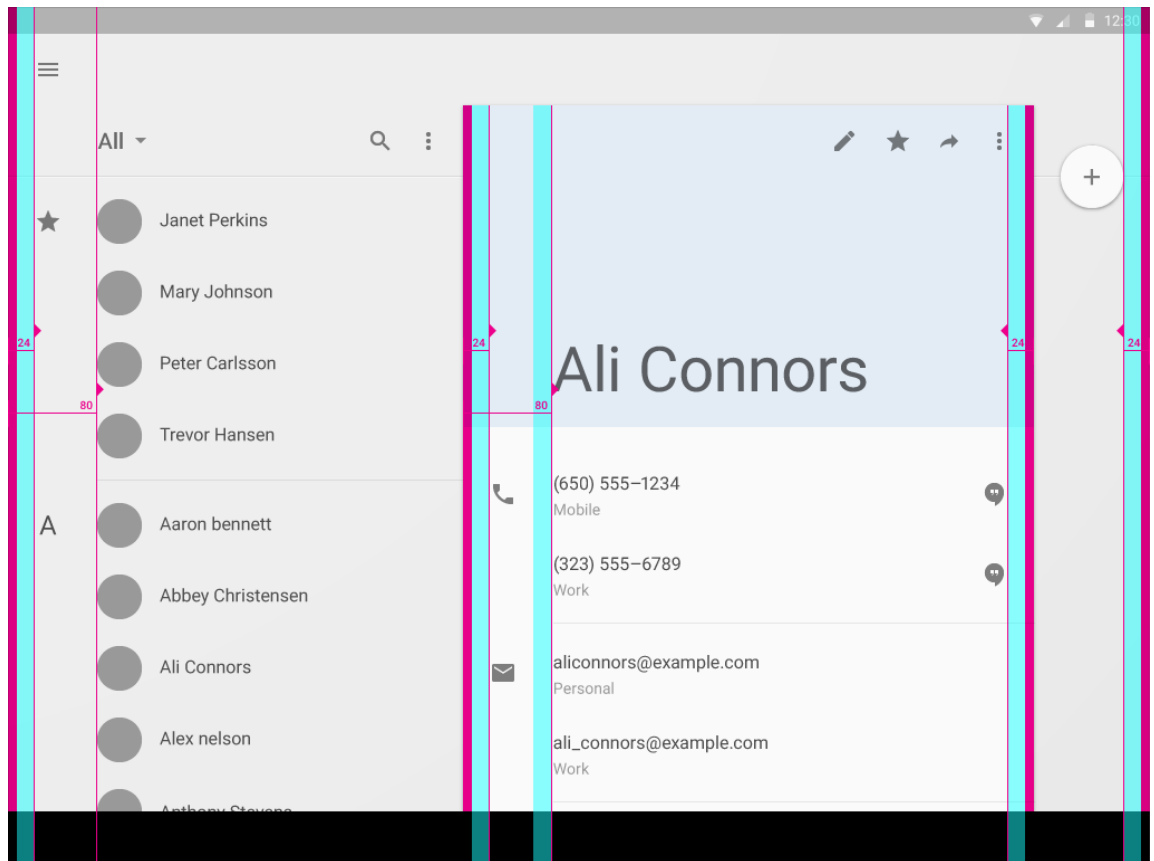
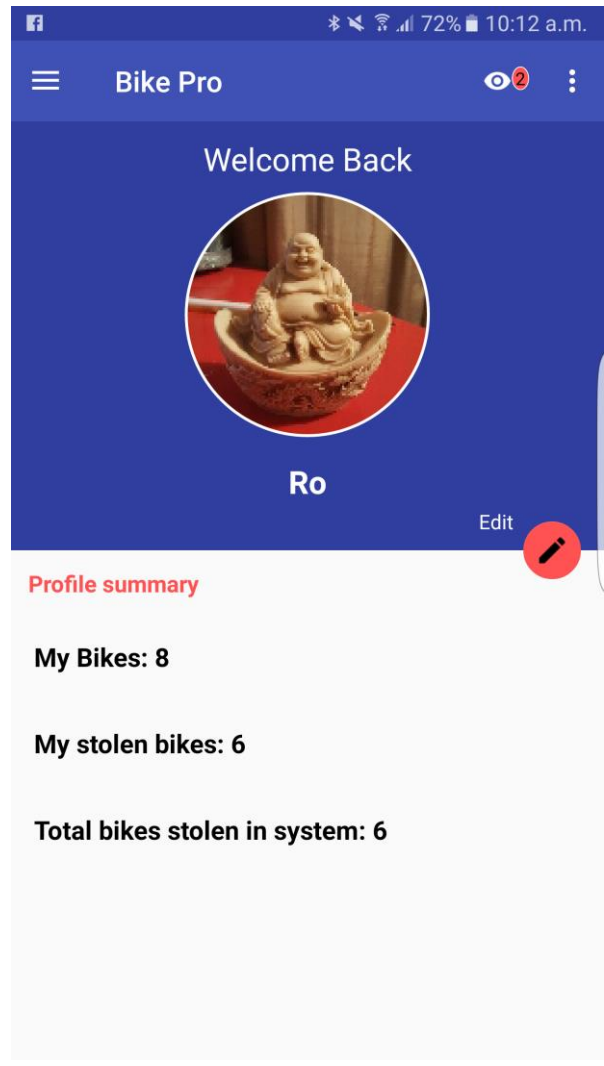
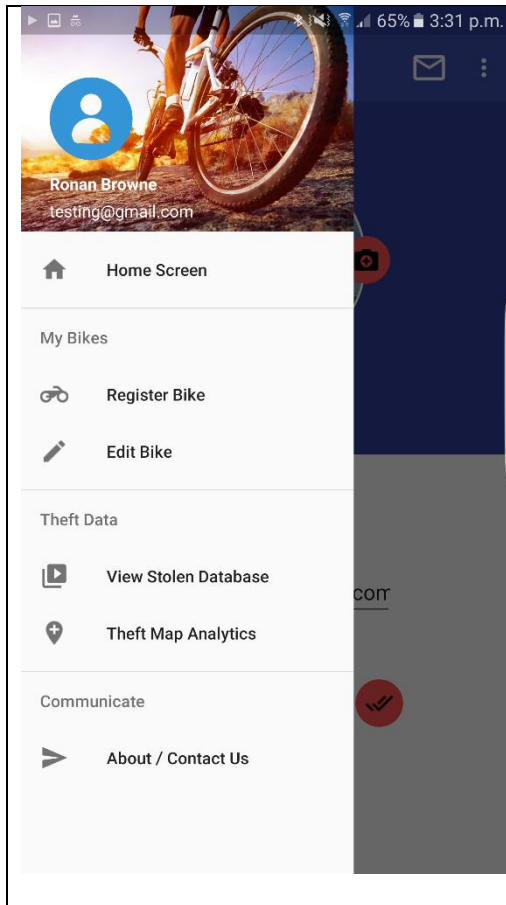


Image 4.5.1 – Image from <http://material-design.storage.googleapis.com/>

The above image is a sample provided by google of how a developer would comply with UI standards, this project will implement all google recommendation regarding material design and user interface layout.

	<p>Here we see the home screen of my application, I have implemented the Material design specification which include use of grid-based layouts, animations and transitions and depth and padding effects.</p>
--	---



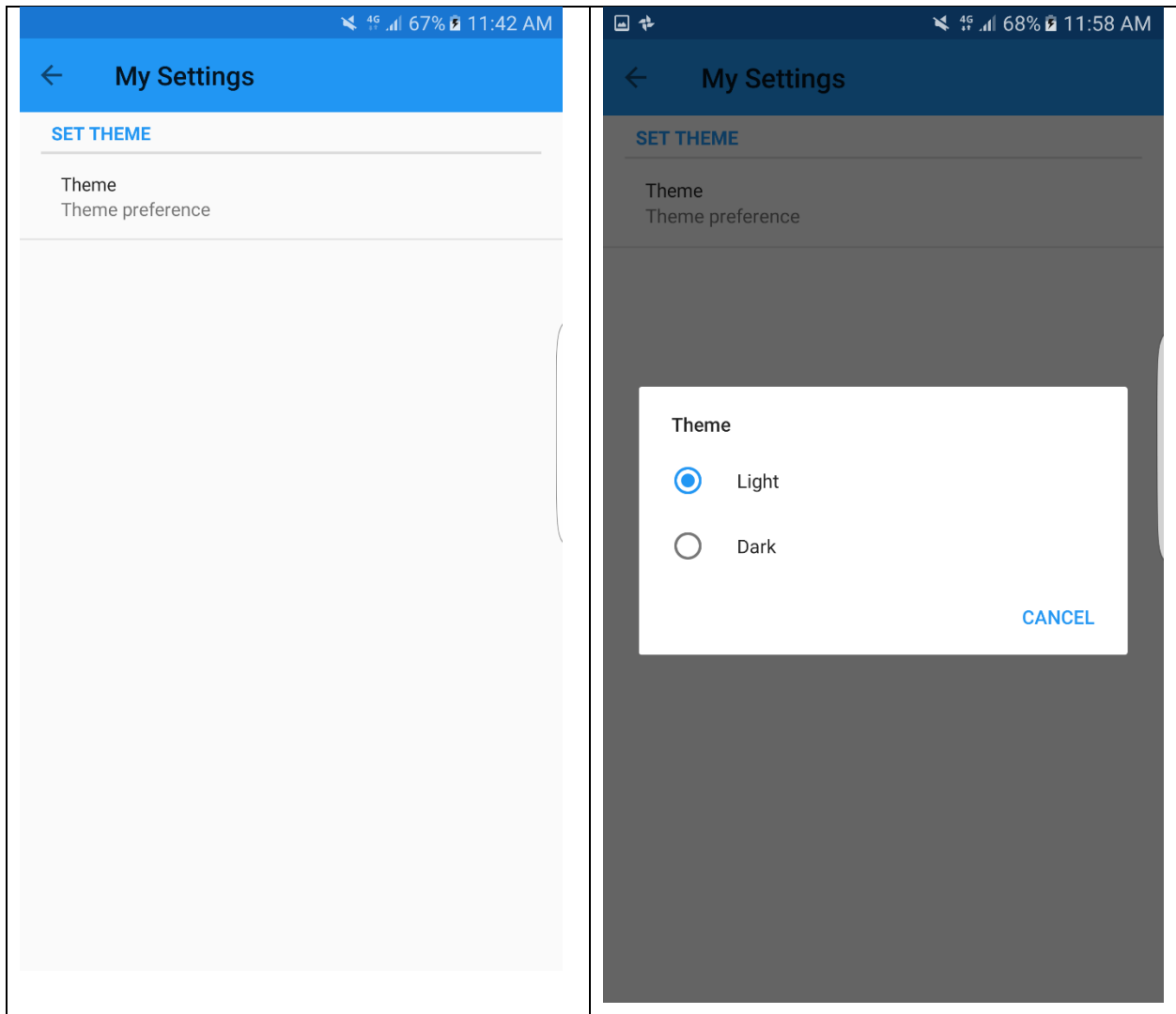


Here we have the navigation drawer Which is the main method for a user navigating in the system. This is available from any screen in the application and is accessed by touching the top left of any screen. It can be dismissed by sliding to the left

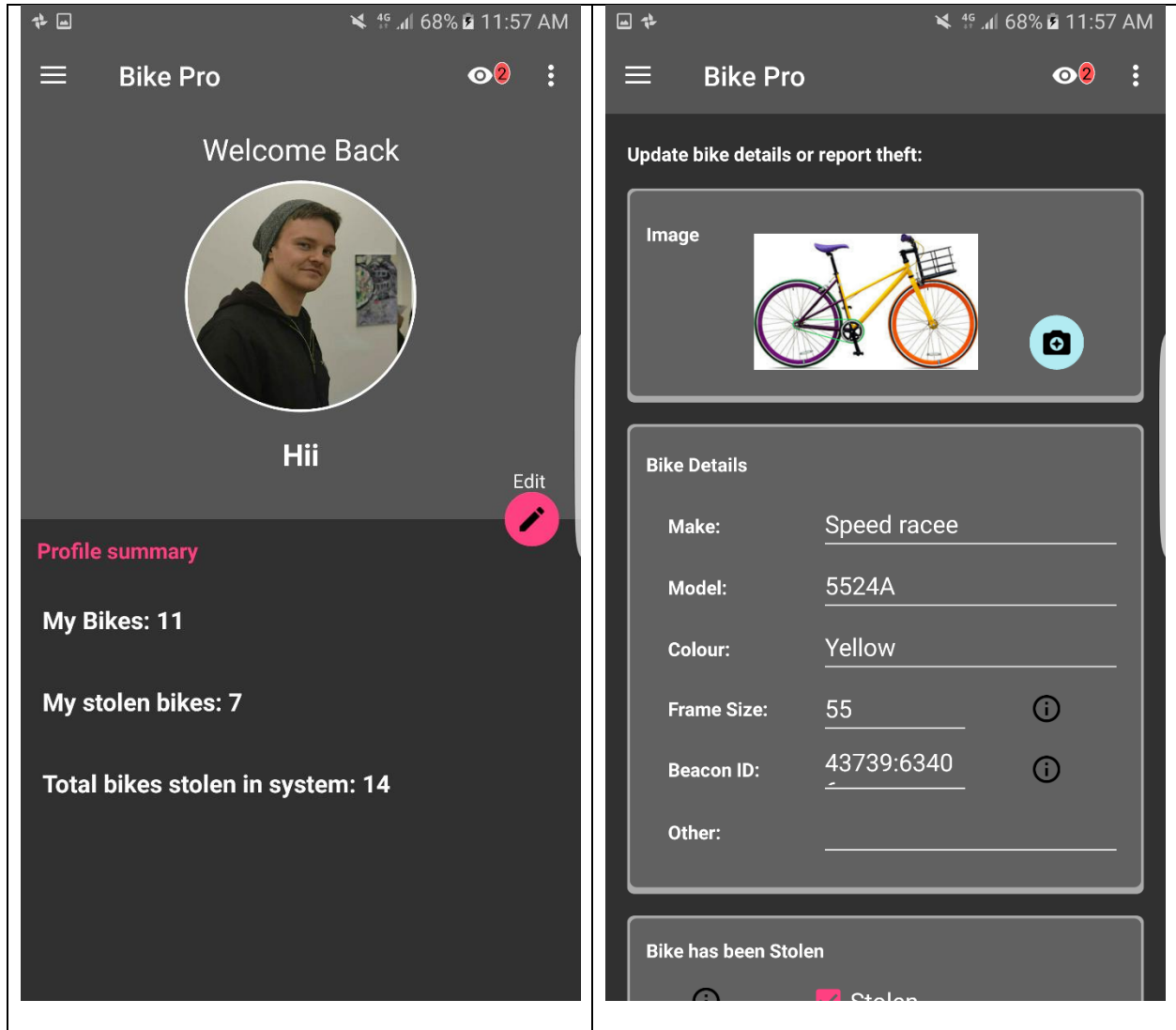
Furthermore, usability is a top priority, regardless if the user is an expert user or a novice they should have no issue navigating and using the applications key features.

The application will have two distinct themes to cater for users with different color preferences or different situations, if using the application indoors at night a user may prefer to use the applications dark theme as this is easier on the eyes.

The theme is changed through the settings menu of the application as shown in the following screenshots.



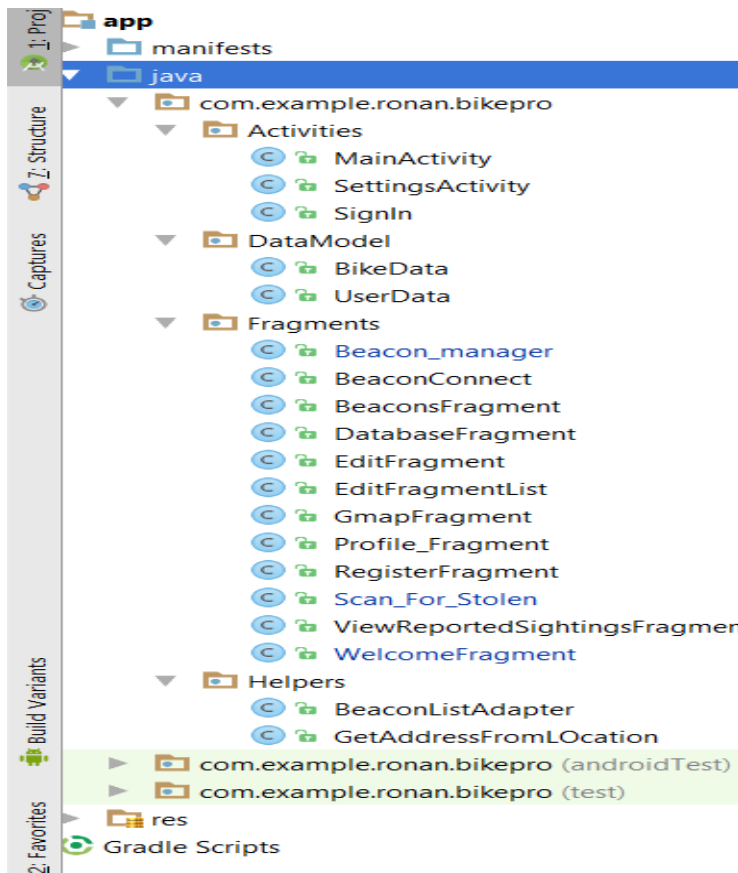
The result of the theme change is then shown in the following two screenshots, this colour change is continued application wide should the dark theme be activated by the user.



## ***2.7 Implementation***

The project shall be well structured, with related classes grouped into parent packages. This shall enable easy finding of classes and ensure any future development of the application is smooth even if taken on by a new developer.

Sample of project Structure:



## **Brief explanation of the role of each class:**

### MainActivity:

Handles the navigation drawer which is viewable from any screen, this application implements fragments, a type of container GUI, every time a user changes screen, that GUI is loaded inside the main activity from the fragment.

### Settings Activity:

This activity is for changing application settings, we change the theme of the application from a light to dark theme via a system preference that the user selects.

### SignIn:

handles sign in features, on opening of the app a user is directed here if they are not signed in otherwise the main activity is called.

### Fragment Package:

All classes here are the various GUI components and their respective logic for example GMapsFragment loads the google maps page which displays a visual representation of all stolen bikes

### Helper Classes:

I employ several helper classes BeaconListAdapter provides the logic to implement a list adapter when returning data found when searching for stolen bikes. GetAddressFromLocation is a helper class which gets the exact street address of the phone using google API location services.

### Interesting code snippets:

What follows are some samples of the more interesting code snippets / features of the application

### Sensor Implementation:

A user will have the option of connecting to a sensor placed on their bike. This connection happens through Bluetooth and uses the Estimote SDK. The following is an example of programmatically connecting to a beacon and monitoring if it is within range. A notification is then sent to the user if it begins to move out of range.

```

// create a new instance of beacon manager class
beaconManager = new BeaconManager(getApplicationContext());
// connect to aspecif beacon
beaconManager.connect(new BeaconManager.ServiceReadyCallback() {
    @Override
    public void onServiceReady() {
        //start monitoring the beacon
        beaconManager.startMonitoring(new Region(
            //UUID is unique identifier for a particular beacon.
            UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"));
    }
});

// Method called when a beacon loses range
beaconManager.setMonitoringListener(new MonitoringListener() {

    @Override
    public void onExitedRegion(Region region) {

        //call method to push a notifiacion to the users device
        postNotification("Unauthorized movement",
            "Check on you're bike!");
    }
});

```

### **Database Query.**

I allow a user to query my database based on location and radius. I use Google maps API to return the results, superimposing a circle on the map based on users input. The user's location they enter is geocoded meaning I extract the latitude and longitude co-ordinated from the given street name in the query.

I then draw a circle overlay on google maps using the radius and provided address. I then only show the user markers where bikes have been registered as stolen within that radius.

This is a screen which can be hidden and the database list is then reset. This fragment also makes use of a sliding animation feature when appearing / disappearing from view.

Method to draw circle on map. Note some functionality such as populating arrays of longitude and latitude used in the following code are handled in other methods not shown.

```
public void drawOnMap(LatLng latLng, int radius) {

    //for debugging
    Log.v("Query paramaters", "Latitude: "+latLng.latitude+"Longitude: "+latLng.longitude );

    //remove previous drawn circle if there is one
    if (circle != null) {
        circle.remove();
    }

    //draw radius of circle on map based on input, this put an over lay of a circle with some shading
    circle = googleMap.addCircle(new CircleOptions()
        .center(latLng)
        .radius(radius)
        .strokeColor(Color.rgb(0, 136, 255))
        .fillColor(Color.argb(20, 0, 136, 255)));

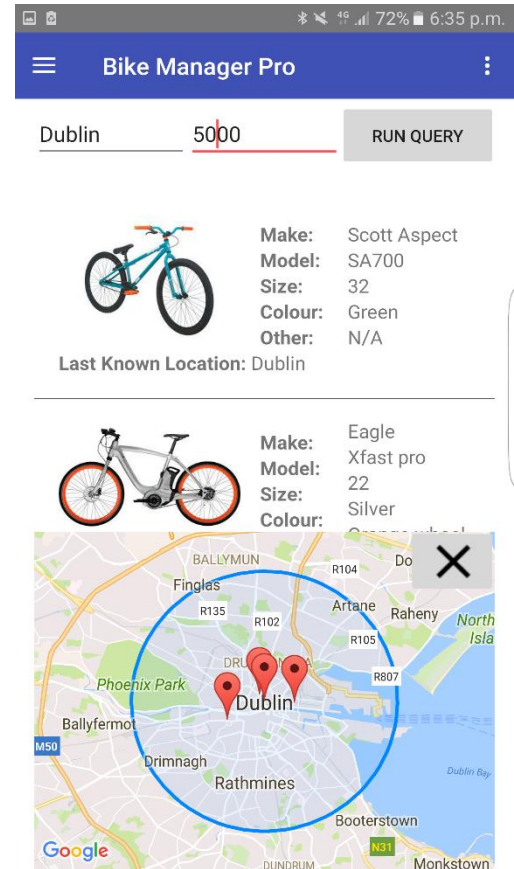
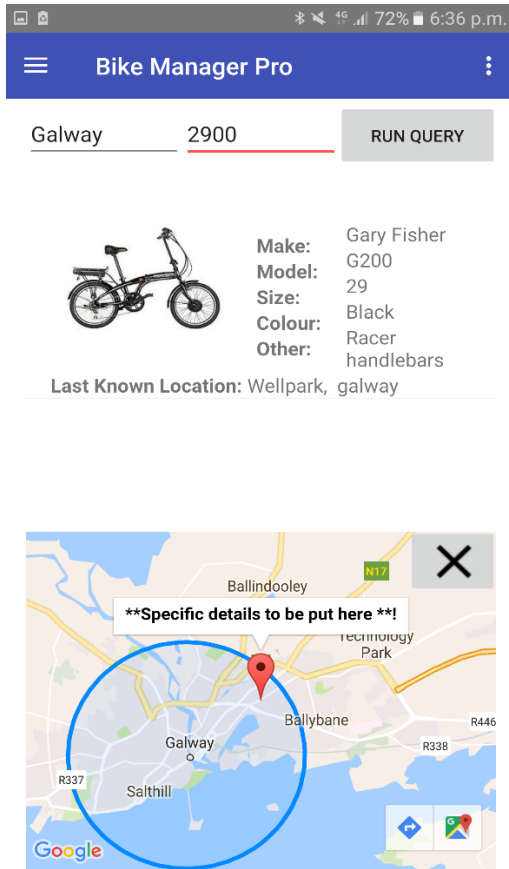
    //create arraylist of markers and co-ordinates that will hold co-ordinates
    List<LatLng> coordinatesList = new ArrayList<>();
    List<Marker> markers = new ArrayList<>();

    //loop through all co ordinates. latitudeArray and longditudeArray are populated in another method.
    for (int i = 0; i < latitudeArray.size(); i++) {
        //create new marker with co-ordinates and add to array list
        coordinatesList.add(new LatLng(latitudeArray.get(i), longditudeArray.get(i)));
        Marker marker = googleMap.addMarker(new MarkerOptions().title("**Specific details to be put here **!")
            .position(coordinatesList.get(i)).visible(false));
        markers.add(marker);
    }//end for

    //i have another method which sets all markers to be false on first laod of map.
    //Here i use the method computeDistanceBetween to see if there are any markers
    // listed in the entered radius and reaveal them if so
    for (Marker marker : markers) {
        if (SphericalUtil.computeDistanceBetween(latLng, marker.getPosition()) < radius) {
            marker.setVisible(true);
        }
    }

    //display markers change the "camera position" of map so it jumps to location user entered
    CameraPosition cameraPosition = new CameraPosition.Builder().target(latLng).zoom(11f).build();
    CameraUpdate cameraUpdate = CameraUpdateFactory.newCameraPosition(cameraPosition);
    googleMap.moveCamera(cameraUpdate);
} //end method
```

Here are two sample screen shots of the previously discussed code snippets in action.



## Scan for stolen bikes in Area

The System allows a user to scan the area using the Bluetooth hardware of their phone to detect any nearby beacons, if these beacons belong to a bike which has been listed as stolen the user will be notified via the user interface, The system will display all details related to the stolen application including last known location, original owner and approximate distance to the bike.

The first screenshot shows a method loops through all the sensors discovered in the area, compares the ID of the sensor to those in the stolen DB and returns a list of matches if they are found.

The second screenshot shows the implementation of this method. In the onBeaconDiscovery method which is part of the Estimote android SDK. Another method is then called which updates the user interface with the data returned



```

//scan the list of beacons returned by ranging and compare to firebase listed as stolen
private List<BikeData> StolenBikesInArea(List<Beacon> beacon) {

    //loop through all beacons returned that are in area
    for (Beacon b : beacon) {

        //get unique Id of a beacon
        String beaconKey = String.valueOf(b.getMinor());
    ;

        //looping through all biked in stolen DB and checking if an these IDs match any stored
        for (BikeData data : bikes) {

            //see if DB stored UUID matches that of one nearby, pull back match if so
            if (data.getBeaconUUID().equals(beaconKey)) {

                //get proximity of beacon
                data.setBeaconAccuracy(Utils.computeAccuracy(b));
                //add to list of stolen bikes in proximity of users phone
                matchedStolen.add(data);
            } else {
            }
        }
    }

} //end main for loop

//return all stolen
return matchedStolen;
} //end method

```

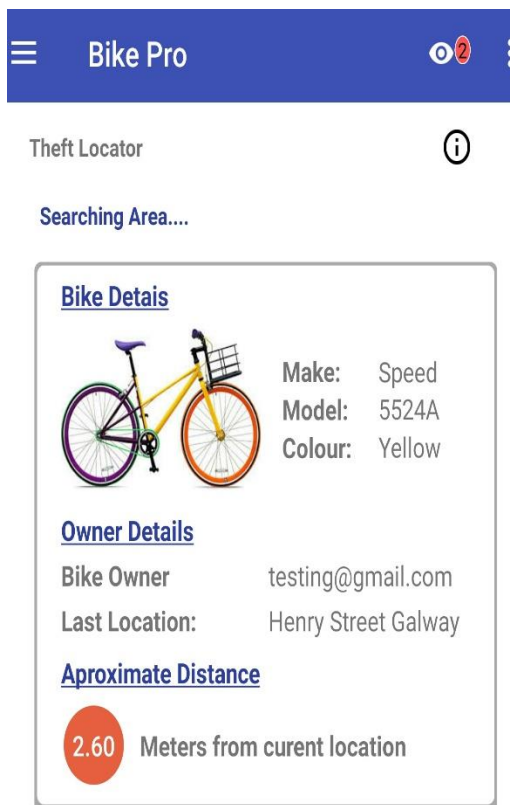
```

beaconManager.setRangingListener(new BeaconManager.RangingListener() {
    @Override
    public void onBeaconsDiscovered(Region region, List<Beacon> list) {
        if (!list.isEmpty()) {

            //return list from method which checks if bikes match stolen DB
            stolenBikes = StolenBikesInArea(list);

            // UI update
            adapter.replaceWith(stolenBikes);
            stolenBikes.clear();
        }
    }
}

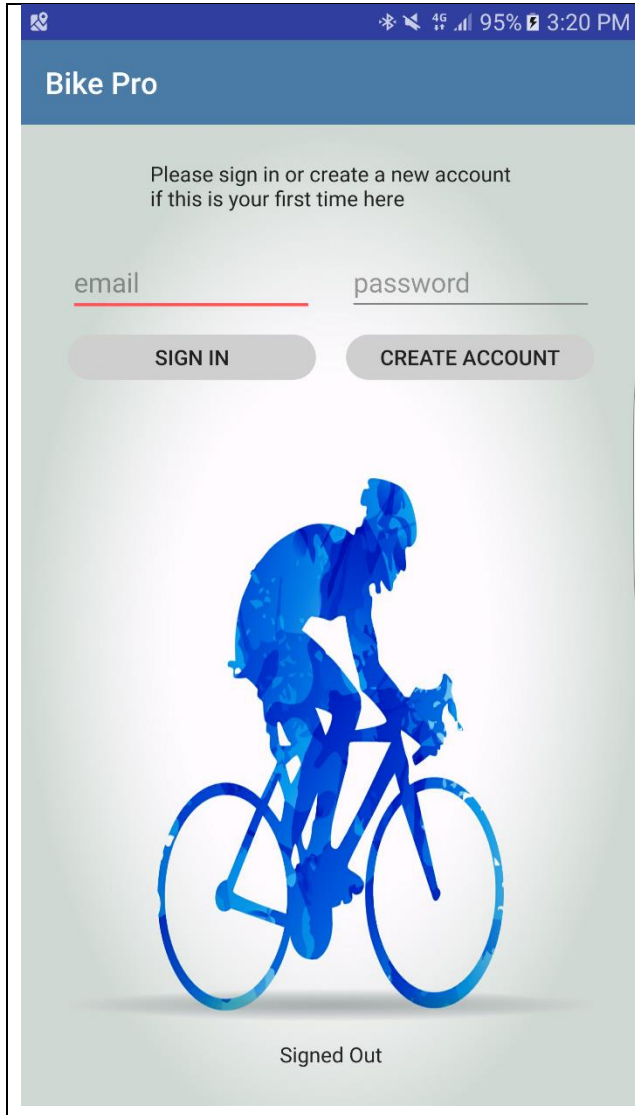
```



Above we see the implementation of the code shown on the previous page to implement the search feature. Here the system has found a bike in the area whose sensor ID matches that of one registered as stolen. Note the red area showing the users distance from the sensor on the bike. This area updates in real time as the user moves, the colour changes also, stronger reds indicating the bike is very close and softer tones of orange and then blue indicate greater distance from the user's phone.

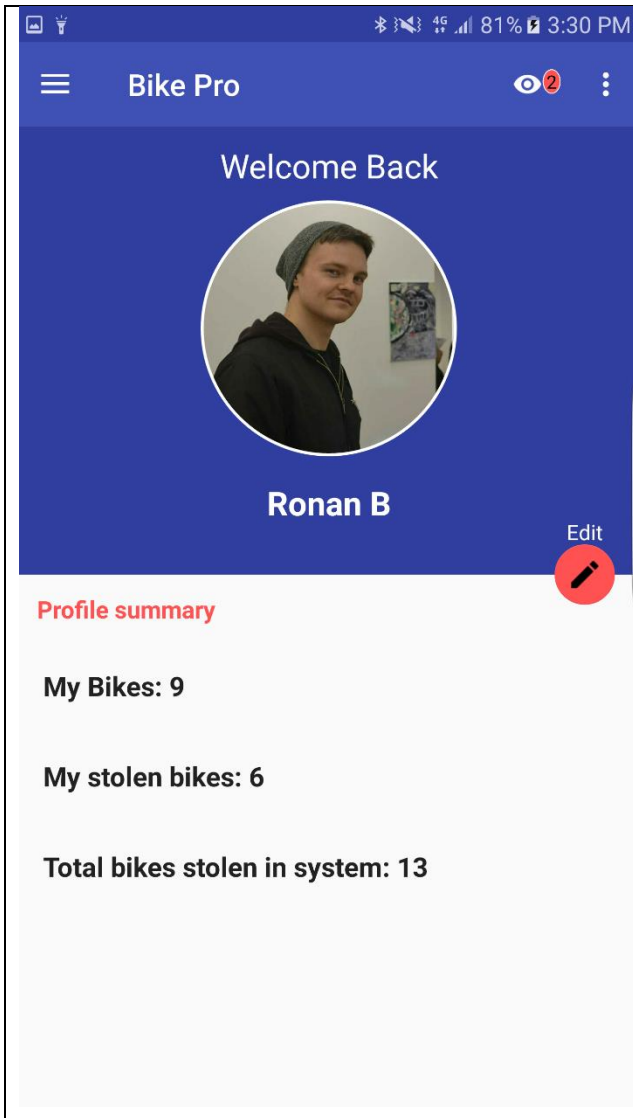
## 2.8 Graphical User Interface (GUI) Layout

In the following section I will provide screenshots and explanations of all screens of the application.



Upon first launch of the application this is the screen the user is presented with.

This is a simple log in screen. A user may log in with an existing account or create a new account from this screen.

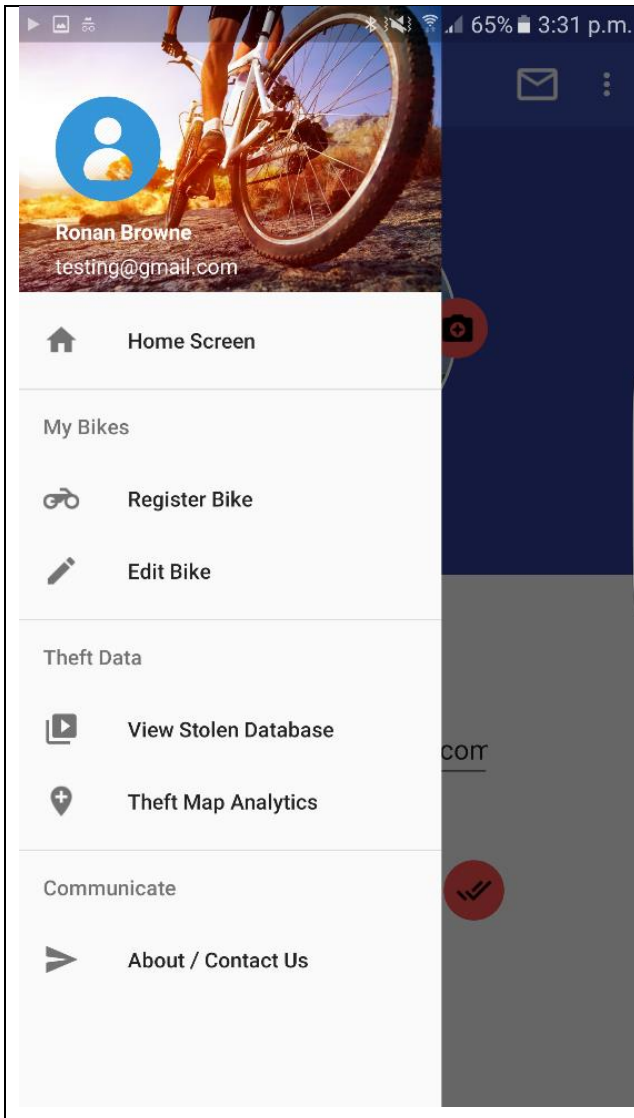


When a user has successfully logged in, they are presented with this home screen. Note in this instance the username associated with this account. On this screen, we provide a quick summary of all information available in the app.

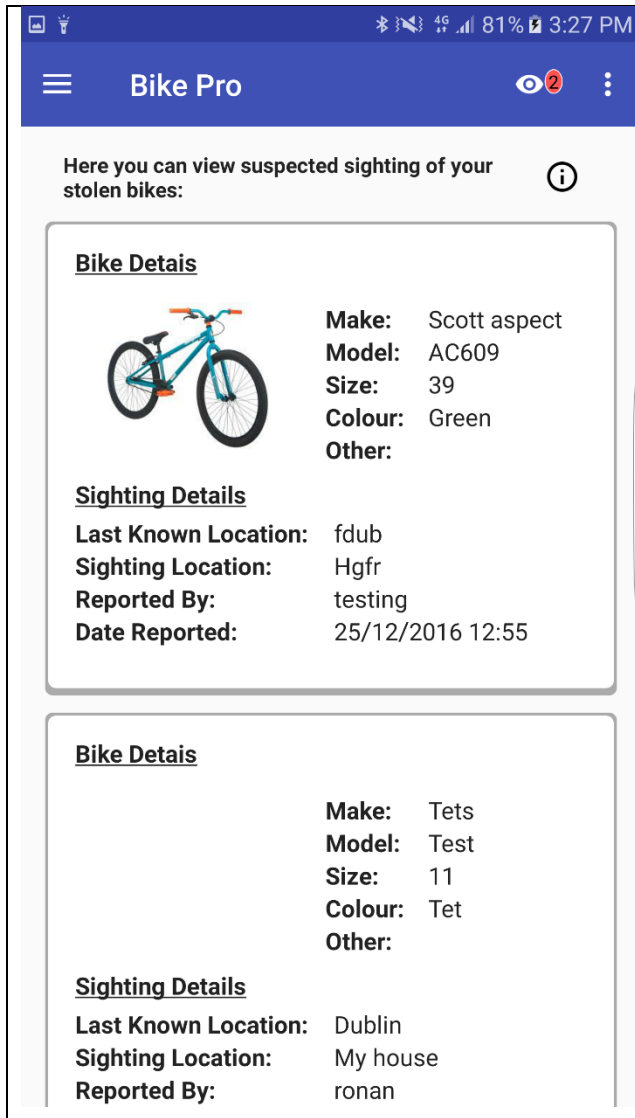
Navigation is handled via the top-left icon, which reveals a sliding menu system.

Also, only available from this home screen, we can navigate to a user's profile page where they may edit their profile details (name, picture).

The envelope icon on the top toolbar of the app will open the user's messaging screen, where they can view if other users have reported any suspected sightings of their bikes logged as stolen.



Here we have the navigation drawer previously mentioned, this is available from any screen in the application and is accessed by touching the top left of any screen. It can be dismissed by sliding to the left

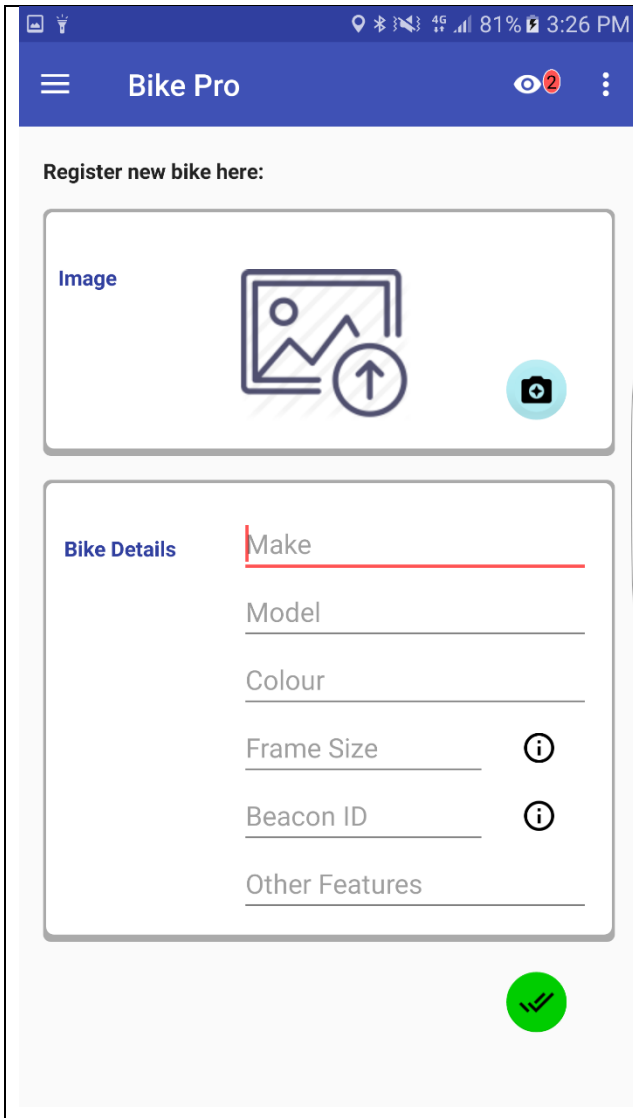


This is the screen which shows users information on reported sightings of bikes they have listed as stolen. This screen is accessed from the eye symbol in the toolbar, note this symbol has a 2 beside denoting there are two reported sightings here.

Example: if user A registers a bike as stolen that bike is then added to the stolen DB. User B suspects they have seen that bike in their area. User B can report a suspected sighting through the application.

All information received about reported sightings is accessed here.

For the purposes of testing this user has many bikes listed as stolen and here we have 2 that have suspected sightings by other users.



Here we have the Register Bike screen, accessed through the navigation drawer.

Here a user will register a new bike with the system, they will upload a picture and input any identifying features of the bike.

This information is saved remotely on the Firebase database

Select Bike you wish to edit:



**Make:** Scott Aspect  
**Model:** SA700  
**Size:** 3  
**Colour:** Green  
**Other:** N/A

**Last Known Location:** Dublin



**Make:** Town Bike  
**Model:** Cruiser '15  
**Size:** 22  
**Colour:** Yellow  
**Other:** Basket

**Last Known Location:** Cork



**Make:** Custom Made  
**Model:** 32a  
**Size:** 6  
**Colour:** Purple  
**Other:** Rear seat

**Last Known Location:** Cork



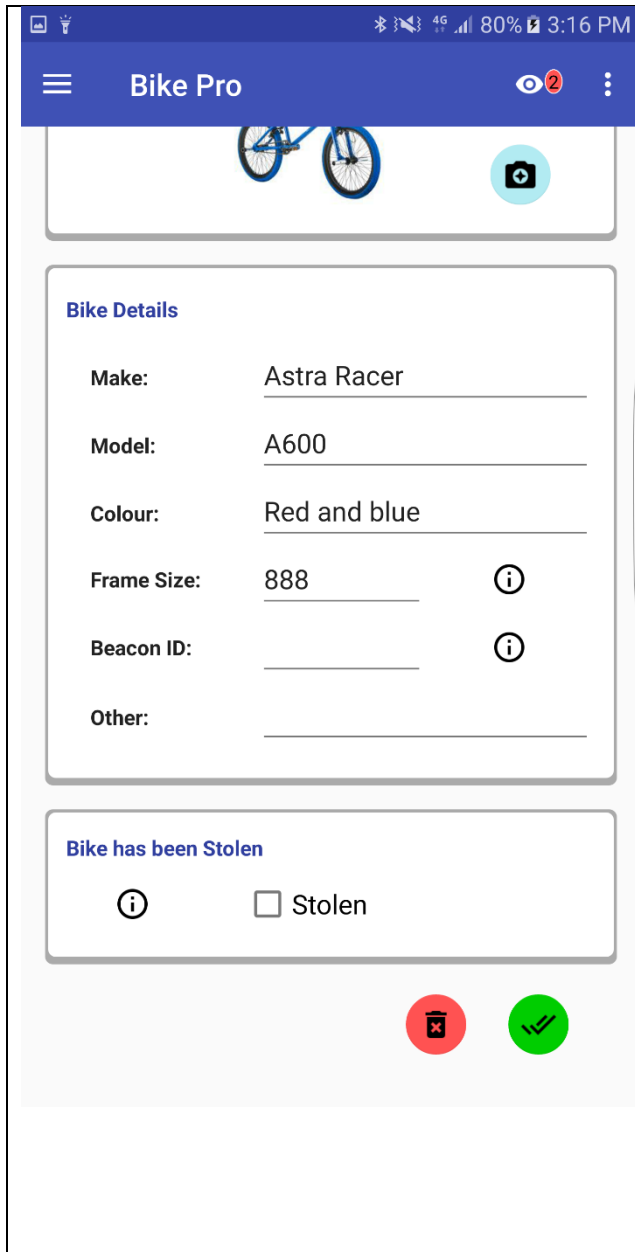
**Make:** Gary Fisher  
**Model:** G200  
**Size:** 29  
**Colour:** Black  
**Other:** Racer handlebars

**Last Known Location:** Wallasey

A user may have one or many bikes registered to them, through the navigation menu once "Edit Bike" is selected a user is shown a list of all bikes they have registered on this account.

From here a user will select the specific bike they wish to edit.





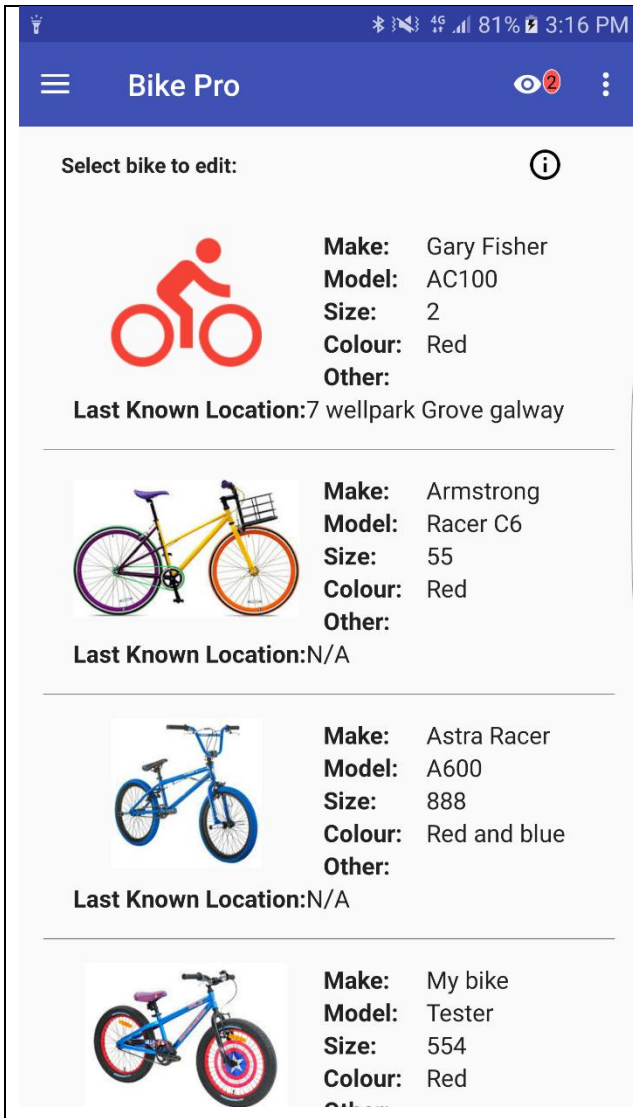
From the previous Edit Bike screen once a user specifies which of their bikes they wish to edit they are brought to this screen.

Here they may edit specific bike attributes, change the picture associated with bike.

Here a user can also register a bike as stolen. Once the stolen checkbox is ticked an area becomes visible which prompts the user to enter a last known location.

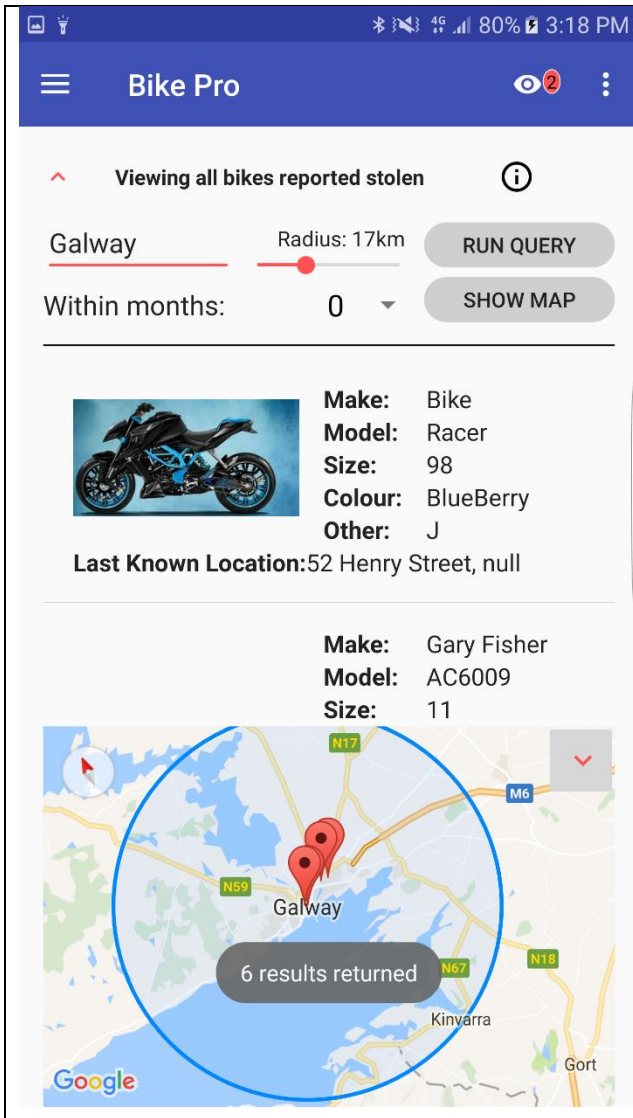
The address entered is geocoded into by the application, meaning we store the latitude and longitude coordinates of the entered address, we use this data elsewhere in the application for working with the Google Maps API.

Once a user click done icon here all data is updated on the DB.



Here we are viewing the Stolen DB screen. Here a user sees all bikes listed as stolen in the DB. This bikes may have been registered by any user.

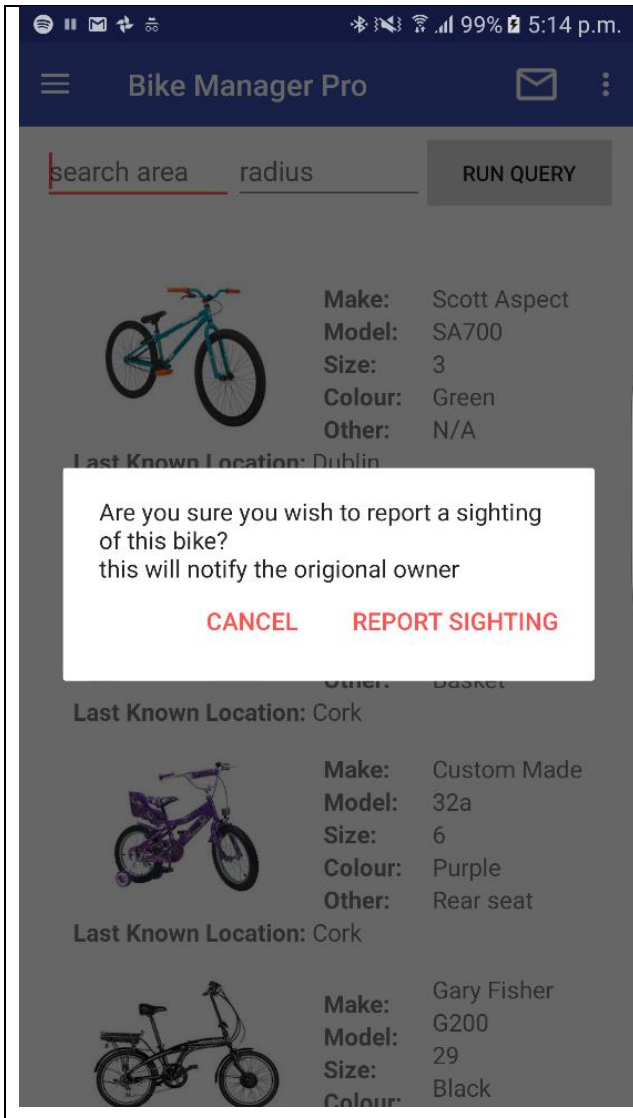
A user can query the DB to make browsing easier.



Here we see an example of querying the DB.

I enter a location and a radius of that location. So this query returns all bikes listed as stolen within a 17km radius of Galway

Also the user gets a visual map representation only showing bikes in the specified location and radius if they click the “show map” button.



While viewing this DB of stolen bikes a user may report a suspected sighting of a bike registered as stolen. This is done by clicking on one of the bikes in this list view.

The user will be promoted to confirm their actions, via a pop up dialog.

Should a user click cancel they will be brought back to viewing stolen DB.

The screenshot displays a list of bikes with their specifications. A modal dialog titled "Sighting Location" is open, showing the text "Mayor's square, Dublin" in a red underline. Below the input field are two buttons: "CANCEL" and "OK".

Bike Image	Make	Model	Size	Colour	Other	Last Known Location
	Scott Aspect	SA700	3	Green		
	Custom Made	32a	6	Purple	Rear seat	Cork
	Gary Fisher	G200	29	Black		

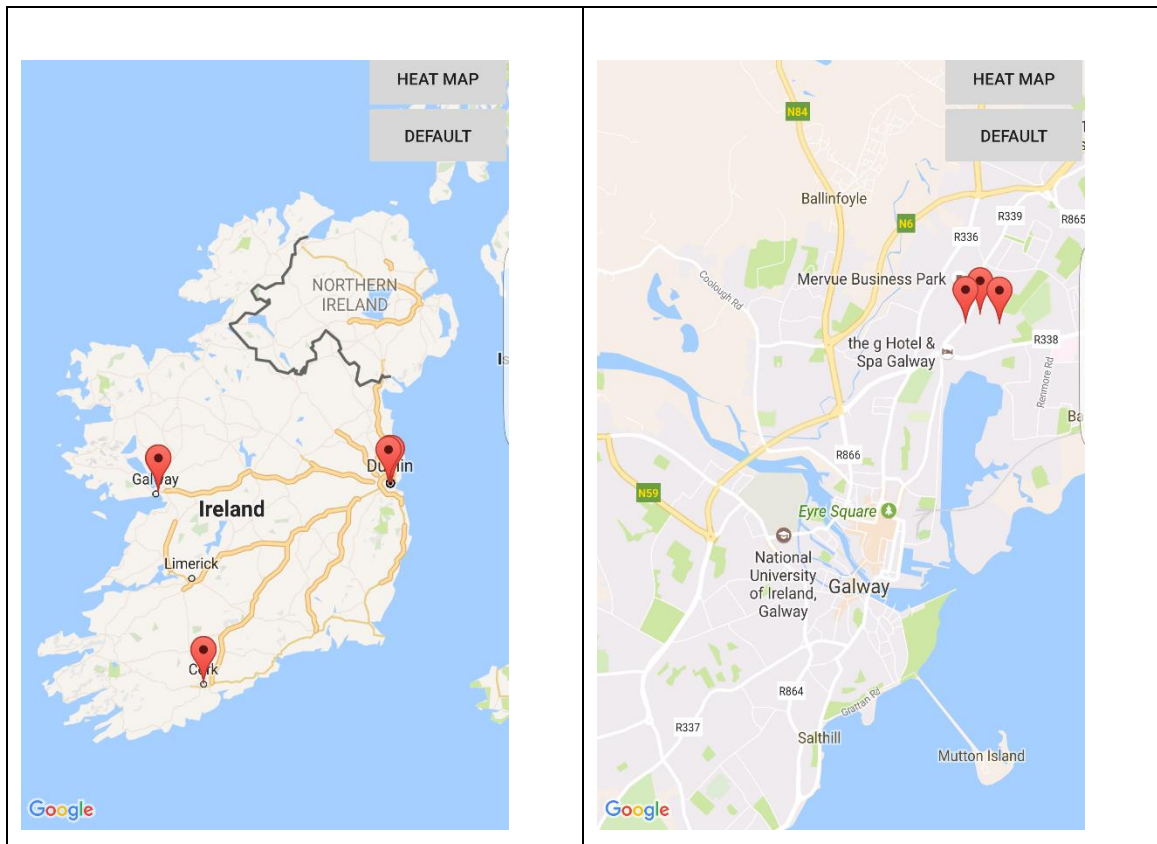
Upon clicking confirm on previous screen the user will be promoted to enter the location of the suspected sighting.

Once a user clicks OK the original owner of the bike will be notified of this sighting next time they use the application.

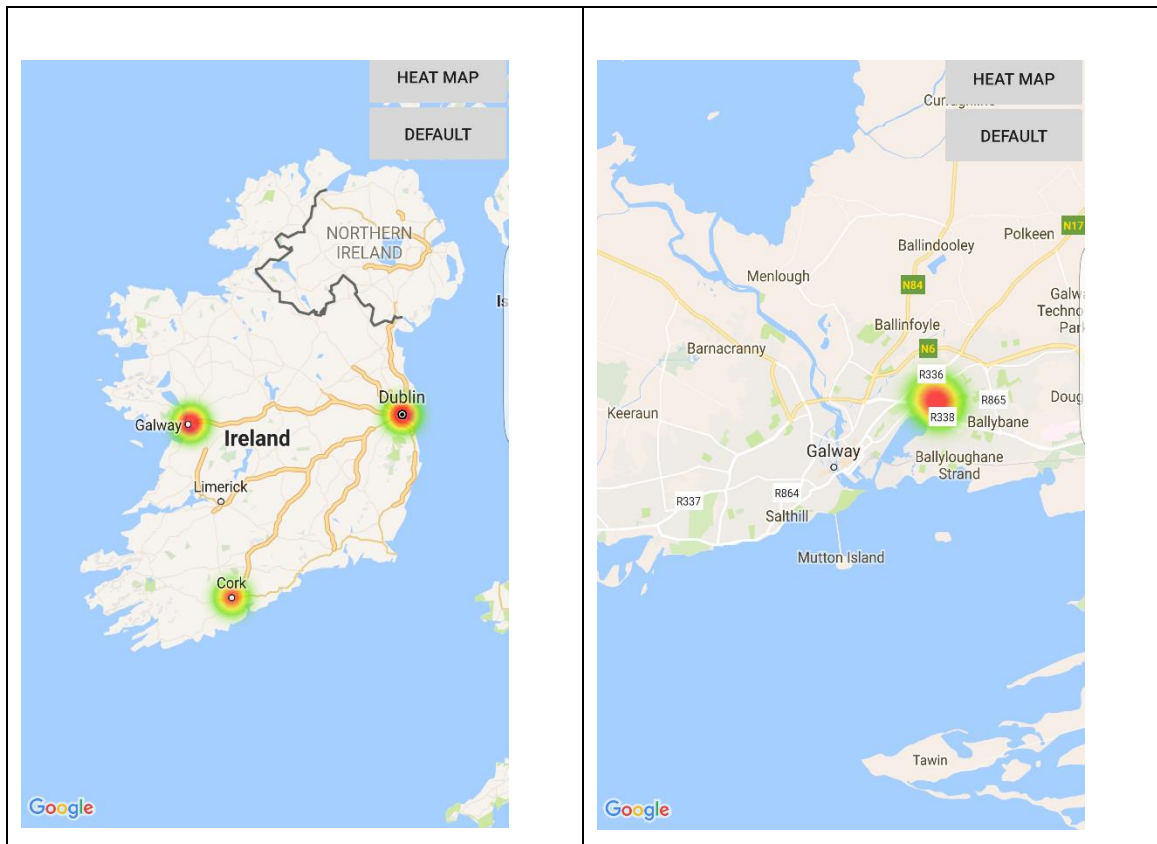
The 4<sup>th</sup> GUI screenshot in this section demonstrates where a user may view reported sightings of their bikes.

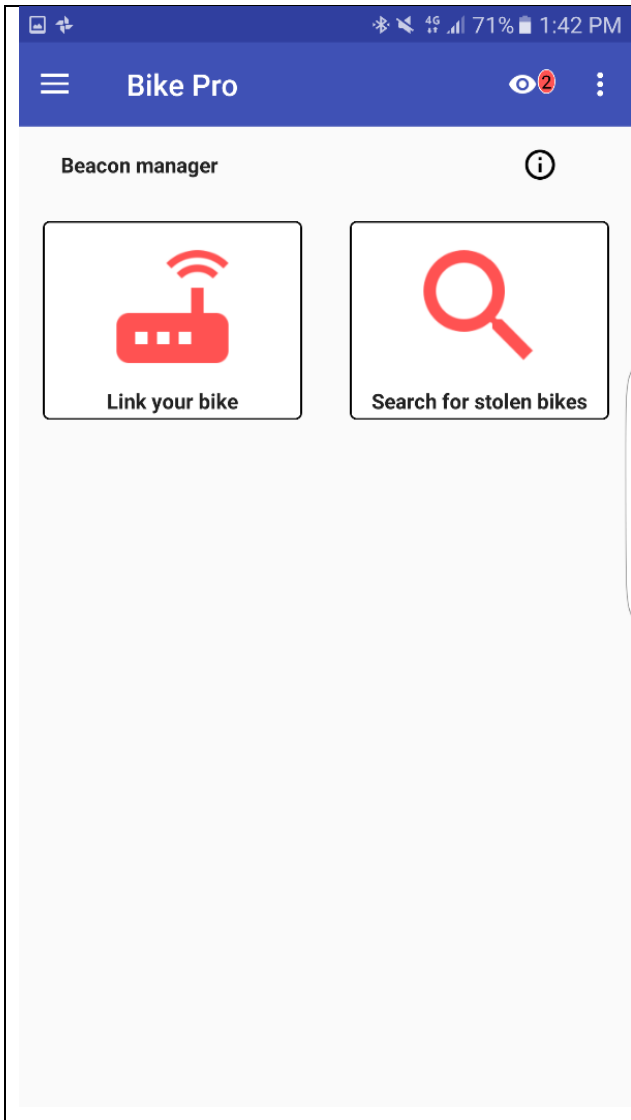
Below we see a sample of the Theft Map Analytics screen. This screen shows data on all bikes recorded as stolen. Here we see a Country wide and then a close in

view. The idea here is over time user data will build up to show theft hotspots and areas best avoided when parking your bike.



Again we have the Theft Map analytics page. This time we implement a heat map setting for quick viewing of high risk areas. Red denoting areas where high concentration of thefts.



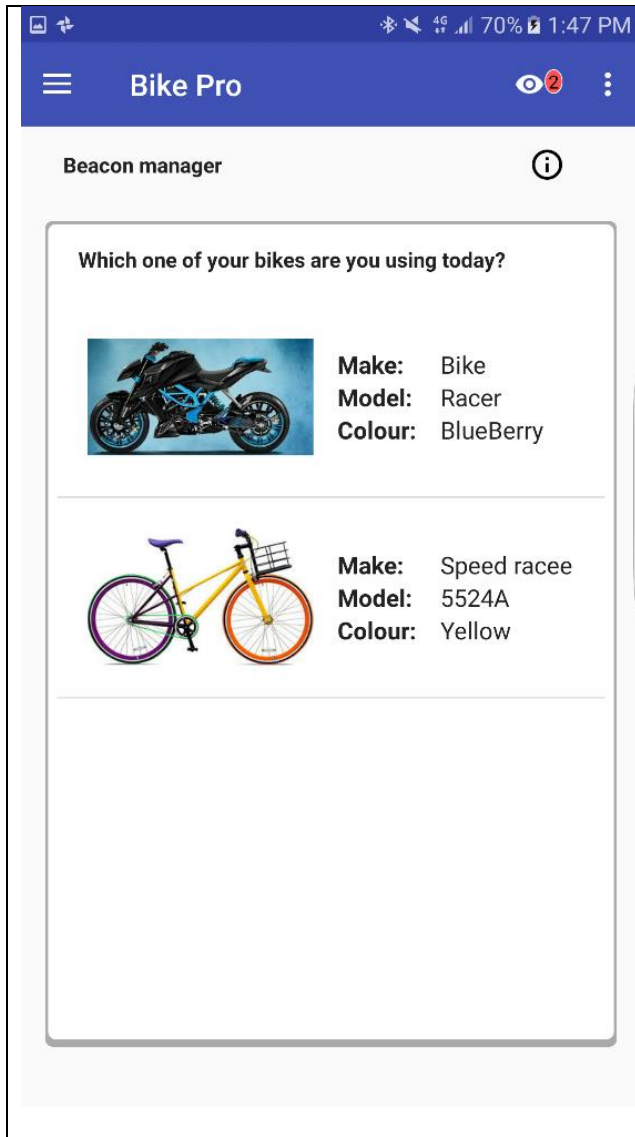


Here we have the "Link to bike" screen which can be activated via the Navigation menu.

Here a user may choose between the two main uses of the sensors.

Link to your bike, which is used to notify the user of unauthorized movement or search for stolen bikes which scans the area for bikes with beacons listed as stolen.

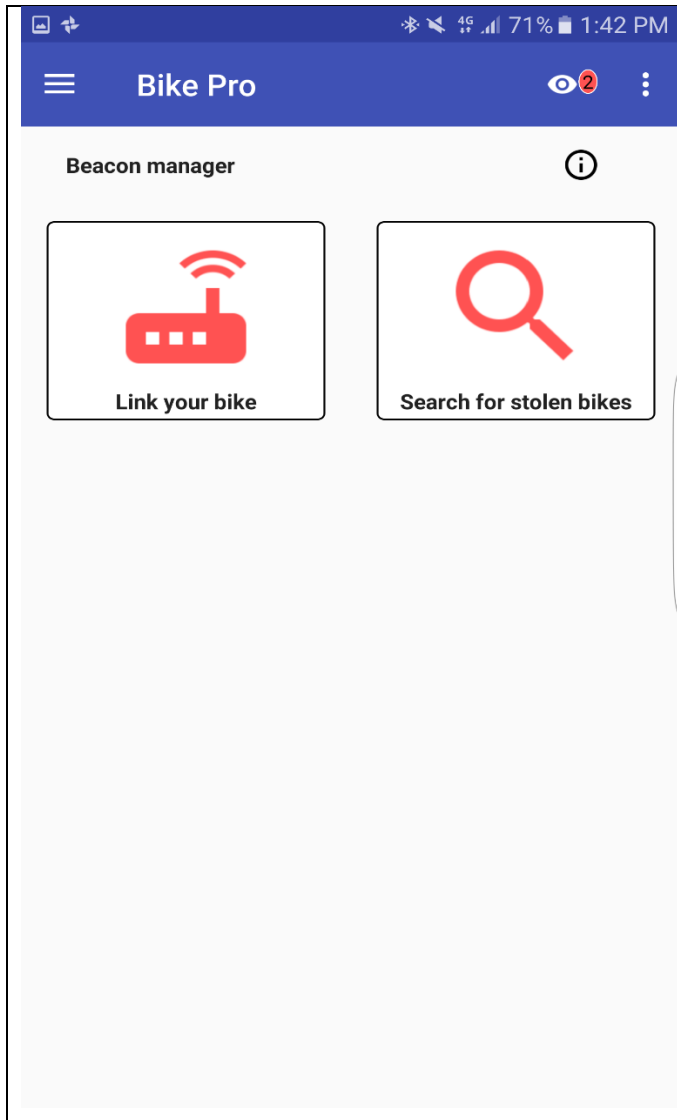




This screenshot shows where a user is taken if they click “link to bike” from the previous screen.

A user may have multiple bikes associated with an account, so if this is the case the user chooses which bike they are using today.

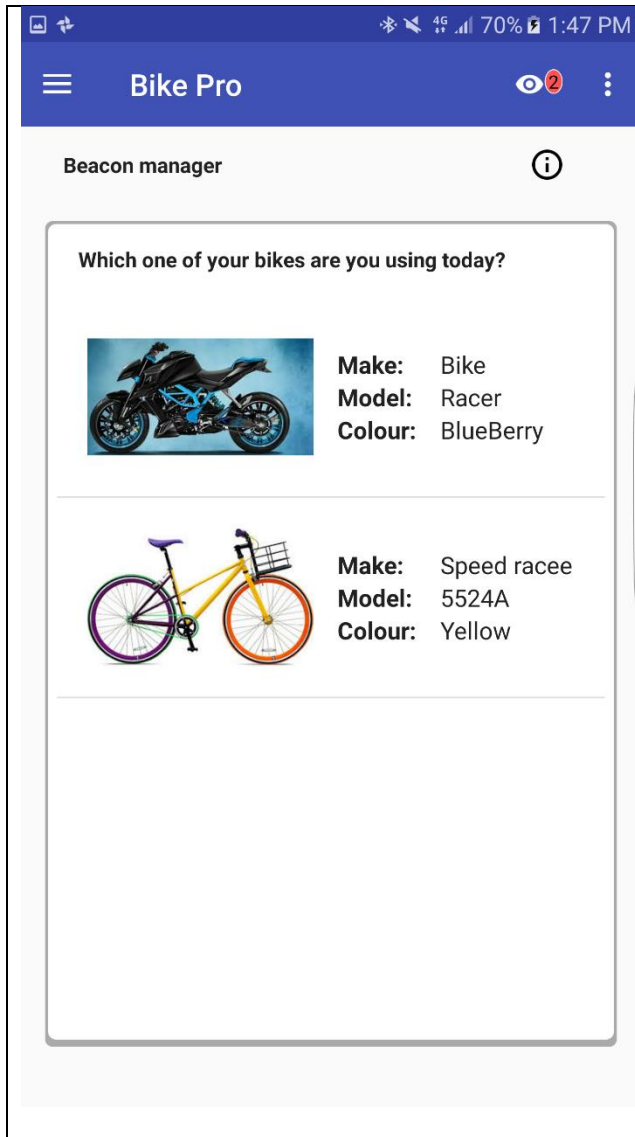
In this case the user has two registered bikes that have corresponding sensors.



Here we have the "Link to bike" screen which can be activated via the Navigation menu.

Here a user may choose between the two main uses of the sensors.

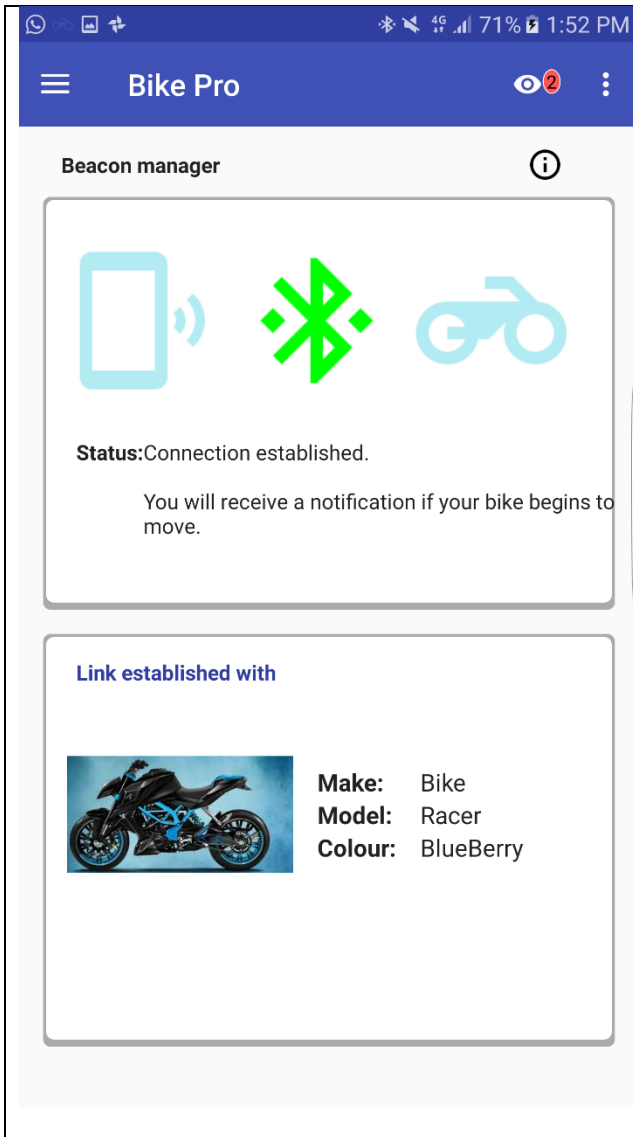
Link to your bike, which is used to notify the user of unauthorized movement or search for stolen bikes which scans the area for bikes with beacons listed as stolen.



This screenshot shows where a user is taken if they click “link to bike” from the previous screen.

A user may have multiple bikes associated with an account, so if this is the case the user chooses which bike they are using today.

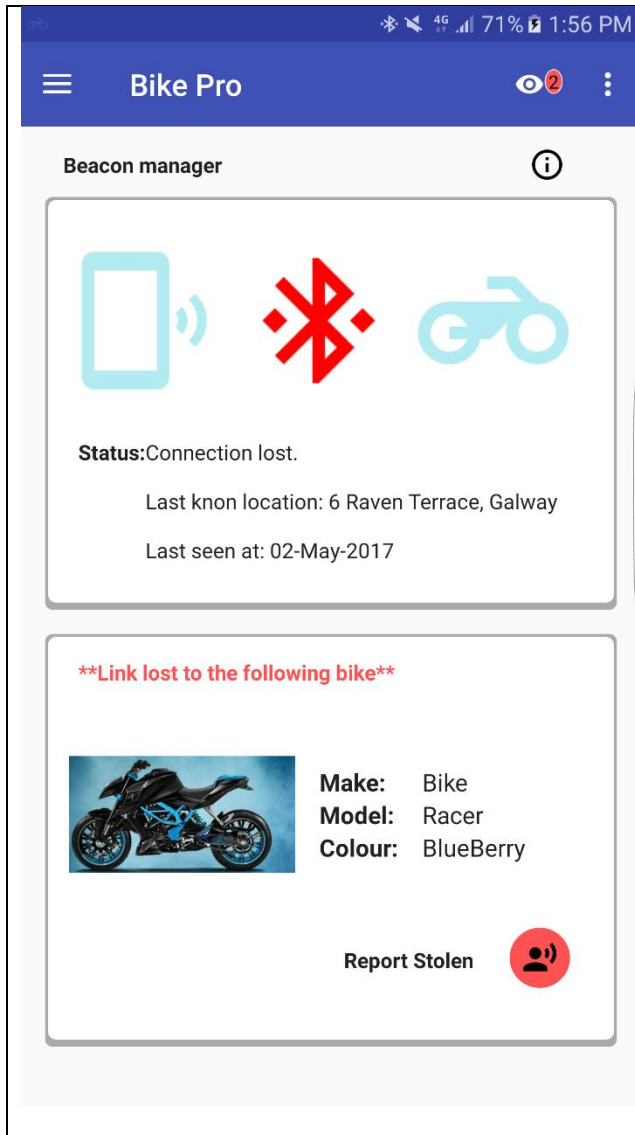
In this case the user has two registered bikes that have corresponding sensors.



Here we see the screen displayed if the user has selected the first bike listed in the above screenshot.

The link has been successfully established as denoted by the green bluetooth icon and the status message.

From here a user will be notified if the connection is then lost.

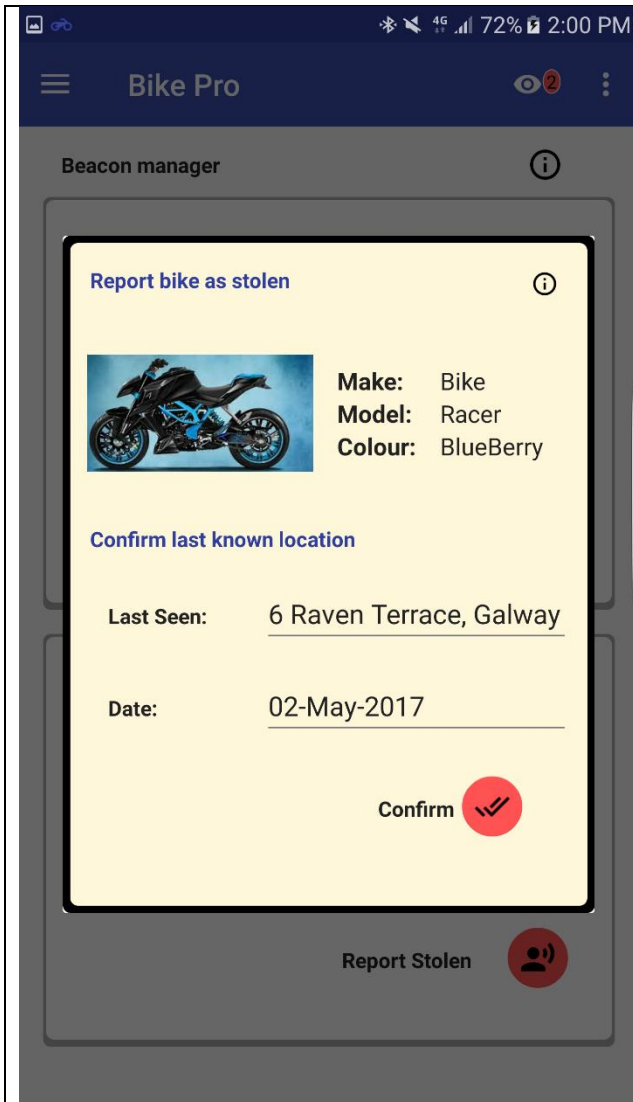


Should the connection be subsequently lost then, by the bike going out of range the user interface will be updated appropriately.

Note the status message has changed from the previous screen also the last known date and location are shown.

A push notification is also sent to the users phone so they will be aware of this even if they are not looking at there phone.

From here a user may also report bike stolen.

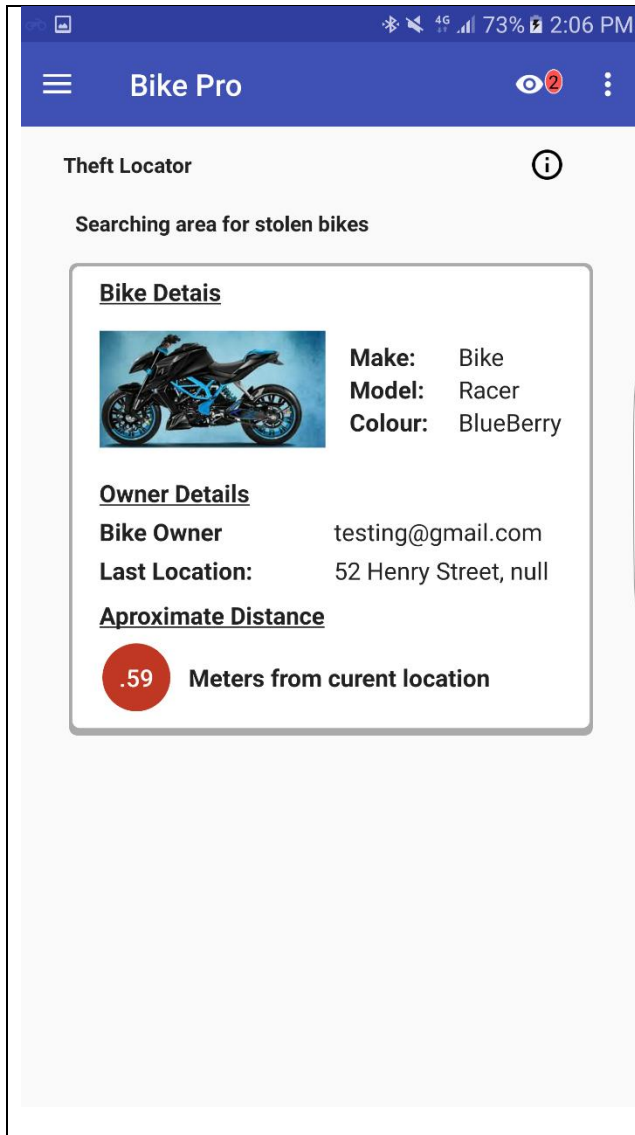


This is the confirmation page to report a bike stolen by using the report button on the previous screenshot when a connection has been lost.

Here a user may confirm if the details are correct, They may change them if they wish.

Location is captured through the google Locations API so it is possible it may be off by a few meters a user may wish to correct this manually.

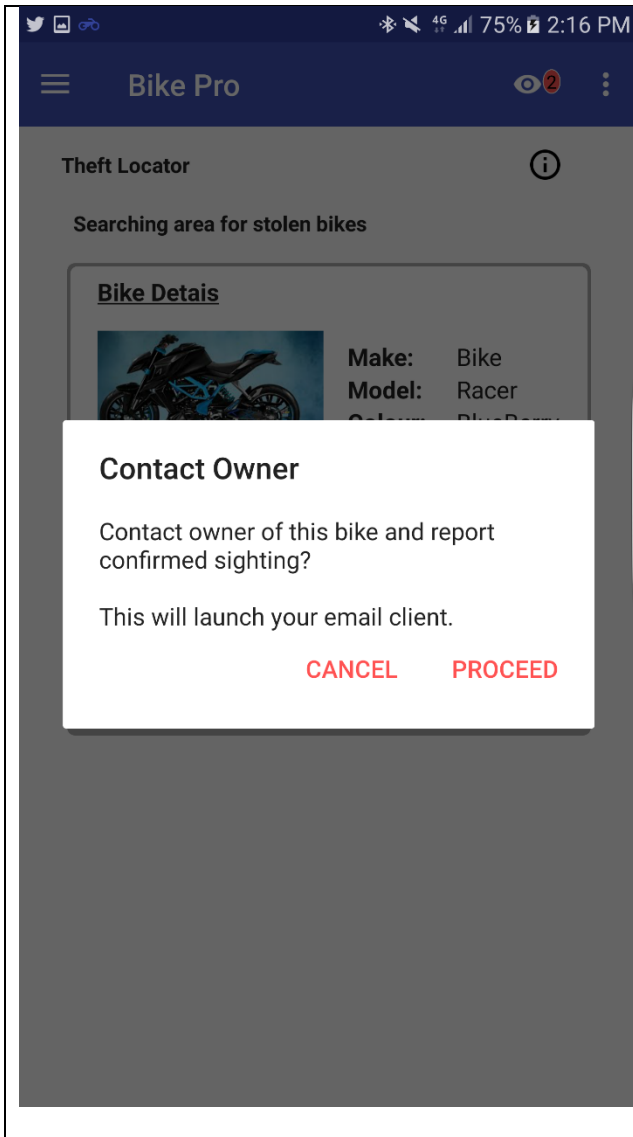
Upon confirming details the bike is added to the stolen database, it can then be viewed in the app under the "view stolen database" section by any user and suspected sightings can be reported.



This is the screen displayed when a user navigates using the main menu to "link to bike" and subsequently selects "Search for stolen bikes".

This feature turns the users phone into a scanning device to look for any nearby bikes with beacons that have been listed as stolen.

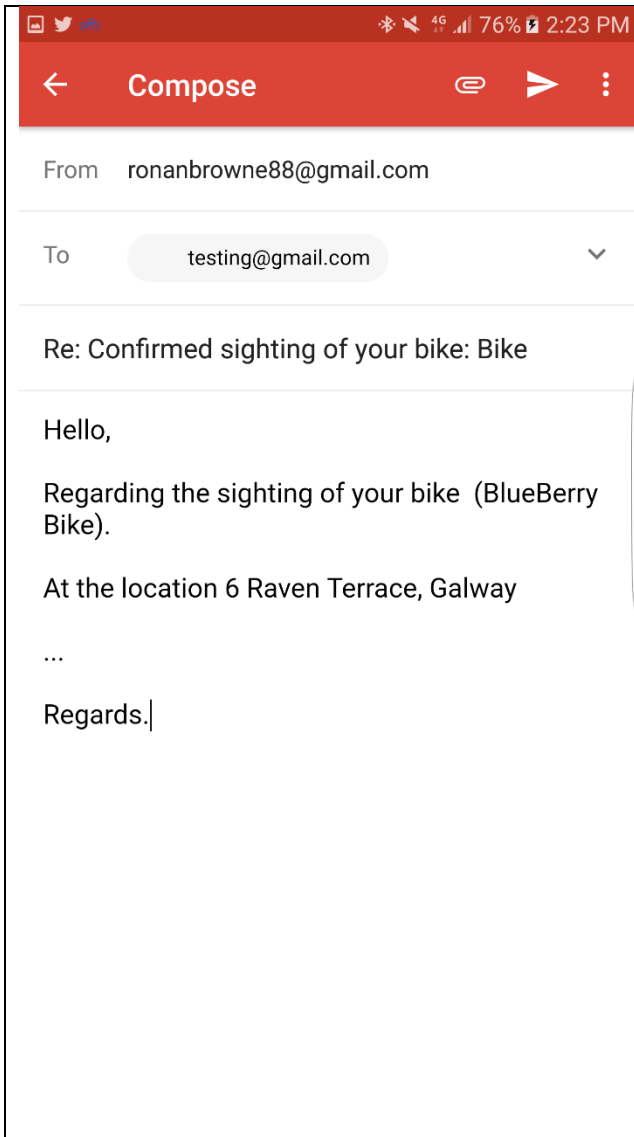
In this case there is one bike nearby, note the distance changes in real time as a users moves further or closer to the bike.



From the above screenshot a user may decide to contact the original owner of the bike once they have identified the location of it in the physical world.

By pressing on the bike in question this launches a dialog to contact the owner via email.





Should the User have pressed proceed in the previous screen there email client will be launched with a pre generated message with the details of the location of the bike.

## **2.9 Testing**

### **2.9.1 Technical Testing**

#### Expresso UI testing

User interface testing will be carried out using googles Espresso Library (<https://google.github.io/android-testing-support-library>). This library will be included in the projects dependency's.

Expresso provides a framework which enables a developer to write classes that will provide automation for testing UI features. Such as testing button clicks and input areas have desired results.

A testing plan will be laid out for each user interface section and the results of each test recorded. The actual result and expected result will be documented. From this process I will eliminate any user interface related bugs or glitches.

Below we have some sample code of espresso tests and their output. The first screen shot shows the code to test the Sign in functionality with a test user.

```

@Test
public void signIN_Test() {
    ViewInteraction appCompatEditText = onView(
        allOf(withId(R.id.field_email),
            withParent(withId(R.id.email_password_fields)),
            isDisplayed()));
    appCompatEditText.perform(click());

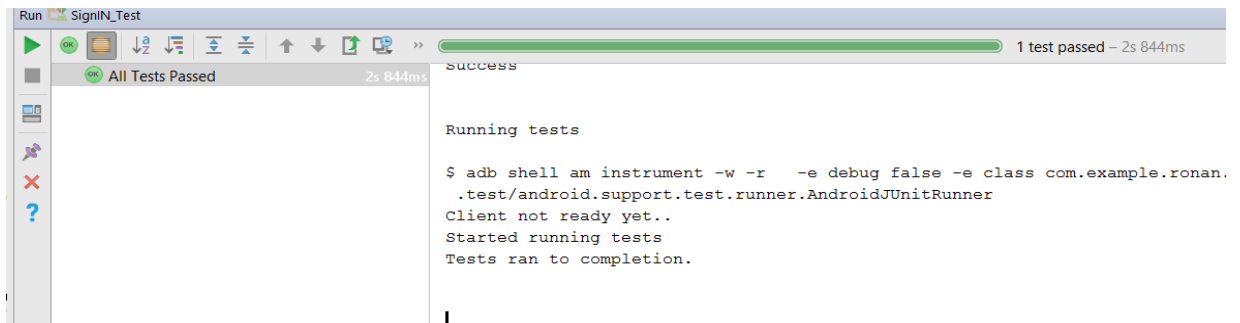
    ViewInteraction appCompatEditText2 = onView(
        allOf(withId(R.id.field_email),
            withParent(withId(R.id.email_password_fields)),
            isDisplayed()));
    appCompatEditText2.perform(replaceText("testing@gmail.com"), closeSoftKeyboard());

    ViewInteraction appCompatEditText3 = onView(
        allOf(withId(R.id.field_password),
            withParent(withId(R.id.email_password_fields)),
            isDisplayed()));
    appCompatEditText3.perform(replaceText("testing1"), closeSoftKeyboard());

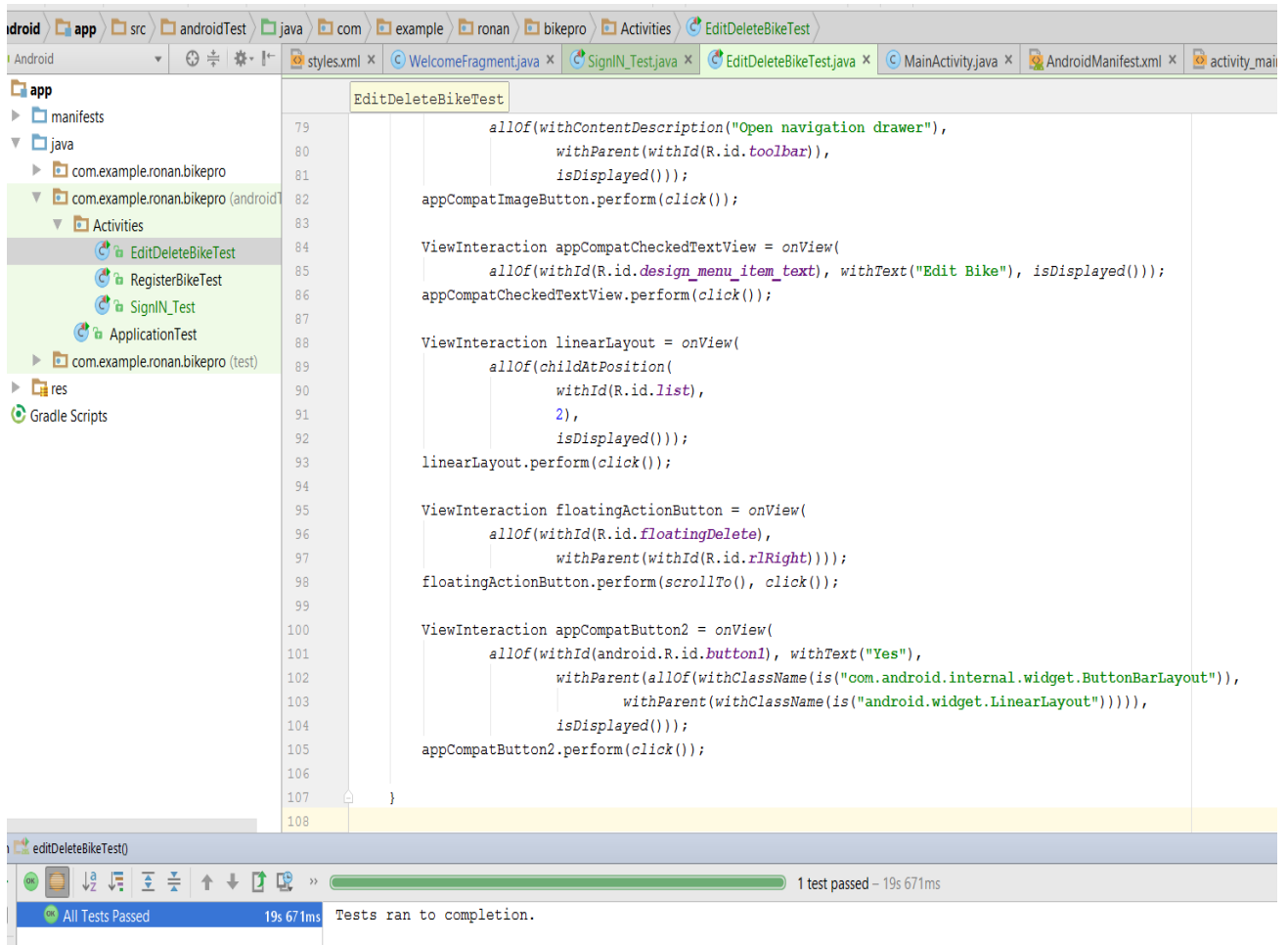
    ViewInteraction appCompatButton2 = onView(
        allOf(withId(R.id.email_sign_in_button), withText("sign in"),
            withParent(withId(R.id.email_password_buttons)),
            isDisplayed()));
    appCompatButton2.perform(click());
}
}

```

And here we see the results of the above test which has passed.



The following screen shot contains a code snippet and out put from the EditDeleteBikeTest.java class which tests if it is possible to sussessfully edit and delete a bike in the syste



### Exploratory Testing:

Manual testing of each feature was carried out by the developer after each new feature has been implemented.

<b>Test Case for exploratory testing of log in feature.</b>
<p><b>Title:</b> Login Page</p> <p><b>Description:</b> A registered user should be able to successfully login at the bike app.</p> <p><b>Precondition:</b> the user must already be registered with an email address and password.</p> <p><b>Assumption:</b> The application has been pre-installed</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"><li>1. Launch application from home screen</li><li>2. In the 'email' field, enter the email of the registered user.</li><li>3. Enter the password of the registered user</li><li>4. Click 'Sign In'</li></ol> <p><b>Expected Result:</b> Welcome page loads showing that particular users overview and they now have access to the full application.</p>
<p><b>Status:</b> passed.</p>

<b>Test Case for exploratory testing of Register Bike</b>
<p><b>Title:</b> Register Bike</p> <p><b>Description:</b> A registered user should be able to successfully Register a bike.</p> <p><b>Precondition:</b> the user must already be registered with an email address and password and be signed in.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"><li>1. Launch application from home screen</li><li>2. Use the navigation menu to select register</li><li>3. Enter all required fields</li><li>4. Select a picture</li><li>5. Press the green confirmation button</li><li>6. User I taken back to the home page.</li></ol> <p><b>Expected Result:</b> Bike is successfully added to the DB and associated with the current user, can now be seen under the Edit bike list.</p>
<p><b>Status:</b> passed.</p>

<b>Test Case for exploratory testing of Editing Bike and listing as stolen</b>
<p><b>Title:</b> Register Bike</p> <p><b>Description:</b> A registered user should be able to successfully edit a bike.</p> <p><b>Precondition:</b> the user must already be registered with an email address and password and be signed in.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"> <li>1. Launch application from home screen</li> <li>2. Use the navigation menu to select edit bike</li> <li>3. Edit the desired fields</li> <li>4. Select the stolen checkbox</li> <li>5. Select a last seen location</li> <li>6. Press the green confirmation button</li> <li>7. User Is taken back to the home page.</li> </ol> <p><b>Expected Result:</b> Bike is successfully added to the stolen DB and is now visible in the “View stolen DB section of app”</p>
<p><b>Status:</b> passed.</p>

<b>Test Case for exploratory testing of Editing Bike and listing as stolen</b>
<p><b>Title:</b> Edit Bike / List as stolen</p> <p><b>Description:</b> A registered user should be able to successfully edit a bike.</p> <p><b>Precondition:</b> the user must already be registered with an email address and password and be signed in.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"> <li>1. Launch application from home screen</li> <li>2. Use the navigation menu to select edit bike</li> <li>3. Edit the desired fields</li> <li>4. Select the stolen checkbox</li> <li>5. Select a last seen location</li> <li>6. Press the green confirmation button</li> <li>7. User Is taken back to the home page.</li> </ol> <p><b>Expected Result:</b> Bike is successfully added to the stolen DB and is now visible in the “View stolen DB” section of app</p>
<p><b>Status:</b> passed.</p>

<b>Test Case for exploratory testing of Viewing all stolen bikes</b>
<p><b>Title:</b> View all stolen</p> <p><b>Description:</b> A registered user should be able to successfully edit a bike.</p> <p><b>Precondition:</b> the user must already be registered with an email address and password and be signed in.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"> <li>1. Launch application from home screen</li> <li>2. Use the navigation menu to “view stolen DB”</li> <li>3. All bikes listed as stolen should be visible</li> <li>4. Run a query</li> <li>5. Select view on map</li> <li>6. A visual representation appears</li> </ol> <p><b>Expected Result:</b> All bikes listed as stolen by any user are shown, a user may query DB and view a visual representation of query data on a map.</p>
<p><b>Status:</b> passed.</p>

<b>Test Case for exploratory testing of viewing Theft Map Analytics</b>
<p><b>Title:</b> View map analytics</p> <p><b>Description:</b> A registered user should be able to view a visual representation of all theft data.</p> <p><b>Precondition:</b> the user must already be registered with an email address and password and be signed in.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"> <li>1. Launch application from home screen</li> <li>2. Use the navigation menu to “Theft Map Analytics”</li> <li>3. Google maps screen loads within app</li> <li>4. All stolen bikes are represented with markers on map</li> <li>5. User clicks heat map</li> <li>6. Stolen bike markers are changed to a hotspots with areas of high theft showing up in stronger colours.</li> </ol>

<p><b>Expected Result:</b> All bike data represented on map via markers or heat spots depending on users preferences.</p>
<p><b>Status:</b> passed.</p>

<p><b>Test Case for exploratory testing of Scanning for Stolen bikes</b></p>
<p><b>Title:</b> Link bike sensor to app</p> <p><b>Description:</b> A user should be able to use their phone to scan the for any bikes with sensors that have registered as stolen.</p> <p><b>Precondition:</b> The user must already be registered with an email address and password and be signed in, with the phones Bluetooth turned on.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p> <p><b>Test Steps:</b></p> <ol style="list-style-type: none"> <li>1. Launch application from home screen</li> <li>2. Use the navigation menu to "Scan for Stolen Bikes"</li> <li>3. Should any bike which have been stolen be in your area they will populate here</li> <li>4. The distance should also appear and change in live time</li> <li>5. Click on a list item will contact the original owner via email.</li> </ol> <p><b>Expected Result:</b> application returns list of stolen bikes in area of phone including info such as last known location, and original owner name and approximate distance calculated through Bluetooth distance.</p>
<p><b>Status:</b> passed.</p>

<p><b>Test Case for exploratory testing of linking bike to app</b></p>
<p><b>Title:</b> Link bike sensor to app</p> <p><b>Description:</b> A user should be able link there bike to the app via remote sensor on the bike and be notified if the bike moves.</p> <p><b>Precondition:</b> The user must already be registered with an email address and password and be signed in, the users bike must have a Bluetooth sensor associated with their bike, phones Bluetooth turned on.</p> <p><b>Assumption:</b> The application has been pre-installed the user is signed in</p>



**Test Steps:**

1. Launch application from home screen
2. Use the navigation menu to “Link to bike”
3. Select the bike you are using today from the list of your registered bikes
4. You should be notified that the connection to the bike has become active
5. Move away from the bike
6. You should receive a notification that the bike is going out of range.
7. A user should have the option of reporting a bike stolen from this point,

**Expected Result:** Application is successfully linked to bike sensor, phone receives a push notification when connection is established or lost, user may register bike as stolen when lost.

**Status:** passed.

### Firebase Test Lab

The firebase platform is not solely a database, it also provides tools for evaluation and testing one of these being Test Lab. Developer can upload their app and the service will run the application on a wide range of virtual devices that emulate various real life phones. Automated tests will take various paths through the application and crashes or unusual activity will be logged. The below screenshot shows testing was successfully carried out on a range of devices.

Robo test, Just now ⓘ

Test execution	Duration	Locale	Orientation	Issues
✓ Nexus 5, API Level 23	–	English, United States	Portrait	–
✓ Galaxy J5, API Level 23	–	English, United States	Portrait	–
✓ Galaxy S7, API Level 24	–	English, United States	Portrait	–
✓ Galaxy S6, API Level 22	–	English, United States	Portrait	–

Summary: Failed 0, Passed 4, Skipped 0, Inconclusive 0

## 2.9.2 Non-Technical testing

### Acceptance Testing:

I performed a demo and received feedback from small user group (friends and family), this was very useful as I received several suggestions which I implemented into my project.

On my welcome screen there was a heading called system overview. One user pointed out “system” is a very technical term. It is something a programmer or IT professional may say regularly but not a term an end user may be familiar with. Based on this feedback I changed this heading to “Application overview”.

I also received feedback on my edit bike screen, a user commented that it was not clear which attribute was which, this prompted me to modify the layout to include an attribute key, beside each text field, such as “Make” or “Model”.

As development continues acceptance tasting will continue with end-users. These test will conform that the system is ready for operational use.

### Five Second Test:

The 5 seconds test is a common testing method in design circles, generally used on web sites the test gets the user to look at the site for 5 seconds and then answer questions on the site, the idea being the user should be able to decide in 5 seconds if they want to stay on your site.

We will adapt this test for use on the mobile application that has been developed. As there are multiple screens and a user will be using a smaller device than a computer screen which the original test was designed for we will expand the test time to 15 seconds.

In total 5 participants took this test.

### Set Up

Participants will only see the application for 15 seconds and they should try and take in as much as they can in that time and try and remember those details. During this time they may use the application as they wish.

After the 15 seconds the phone will be turned off and they will be asked to give honest feedback on their initial thoughts either verbally or in writing.

After they had written down their thoughts some basic questions based on what they saw.

## **Results:**

The following questions were asked to participants based on viewing the application in the allotted amount of time.

### **What is the purpose of this app?**

All users polled were able to correctly identify that the applications main purpose was bike recovery and theft prevention.

### **What did you like most about the design of this application?**

80% of participants said they like the layout and design of the application the color scheme was friendly and the menu system seemed intuitive.

### **What did you like least about the design of this application?**

2 participants mentioned the welcome screen seemed slightly cluttered in the lower half of the screen. While overall the design was parsed users suggested more use of images and graphics to give a more professional feel to application.

## **General Feedback:**

Based on the participants written thoughts after our version of the 5 second test, it was evident that all participants were able to identify the purpose of the application. Overall the feedback was positive and the only negative comments were related to layout and design. As a result of the feedback from this test more polished images will be implemented into the application in the next release cycle.

## **Trunk Test**

The trunk test is a usability test, that states users should be able to quickly determine where they are within a site or app based on content & visual clues. A user should never be “lost” in the system with no obvious ways to navigate home.

### **Set Up**

Users should be able to know where they are within the application based on content & visual feedback. Participants were presented with one of our applications pages and just from the available information on the screen asked to could the identify the following

- What application is this?
- The is the page displayed
- Local Navigation (What are my options at this level?)

For this test we showed users the add bike page and the beacon manager page.

### **Results:**

- What site is this?

All Users were able to quickly Identify the name of the application “Bike Pro” as the name is consistently visible on the toolbar regardless of screen the user is on.

- The name of the Page?

Again all users were able to correctly Identify the page they were on due to the sub heading used at the top of each page

- Local Navigation (What are my options at this level?)

80% of participants to this correctly pointed out the button to open the vertical sliding navigation menu.

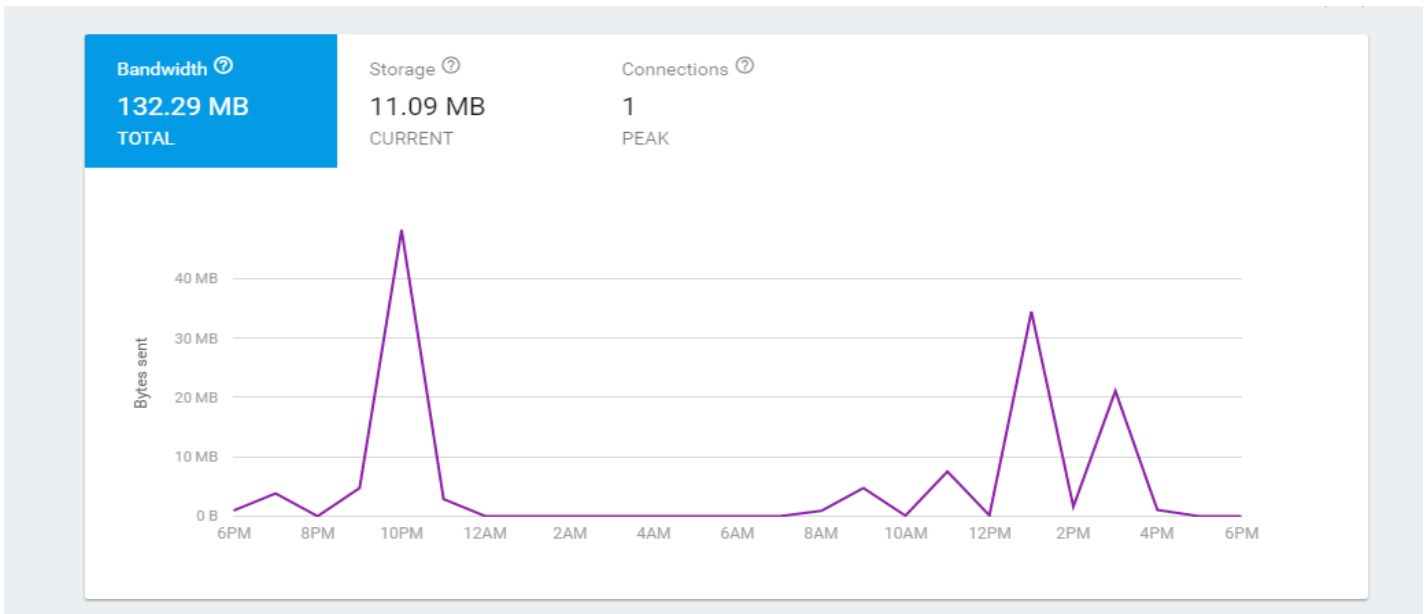
### **General Feedback:**

The application passed the trunk test. On all screen is was obvious to the majority of users where they were in the system and how to reach the navigating menu.

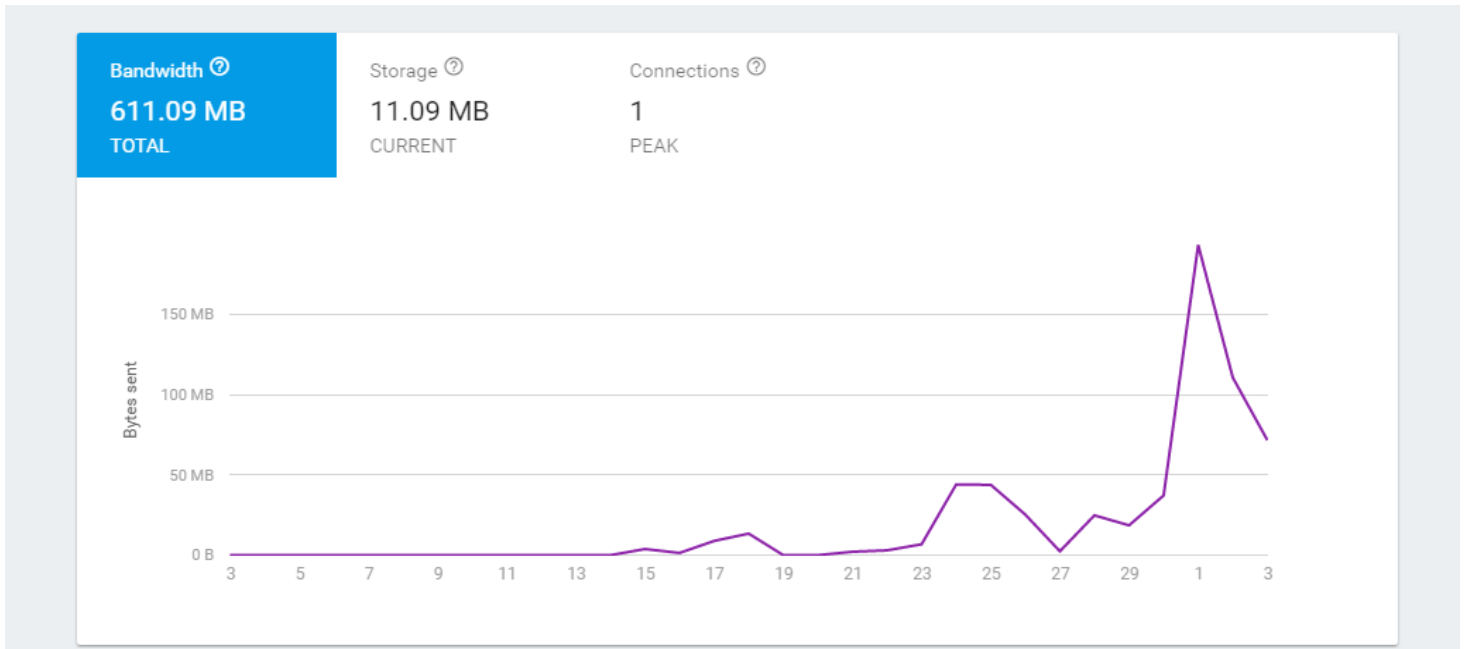
### ***2.10 Evaluation***

System evaluation was carried out using Firebase. The online database also provides tools to monitor usage The following are samples of 24 hours and 30 days of usage outlining trends in active connections, bandwidth used and peak times.

## 24 hours



## 30 days



## Active users and revenue generated over 30days.



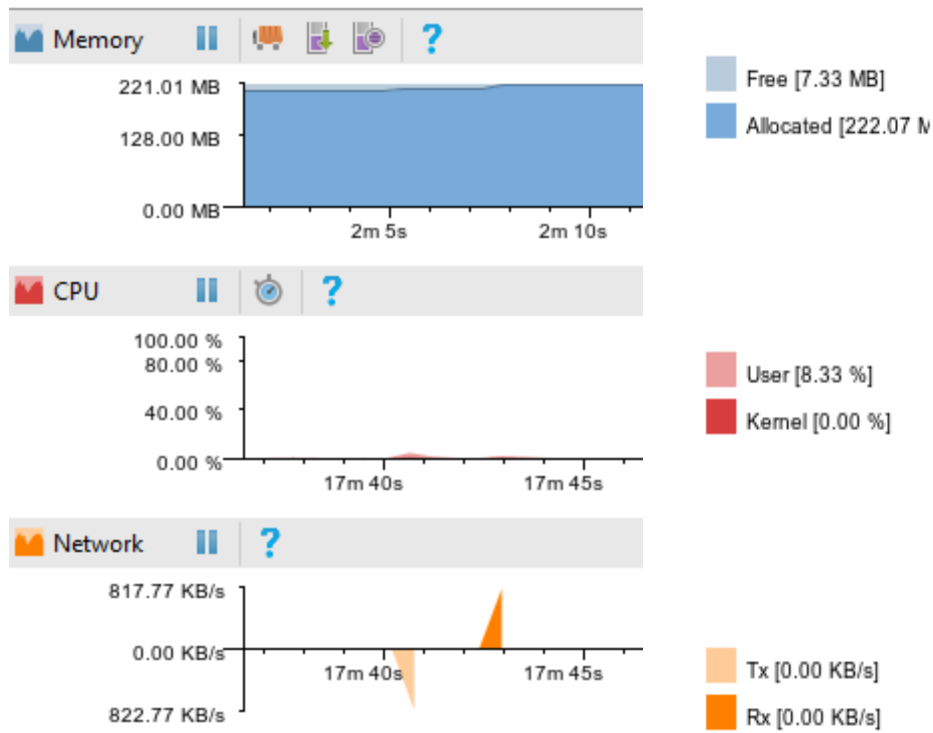
### 2.11 PERFORMANCE

The following images show a measure of system performance under the headings of Memory usage, CPU usage and Network usage.

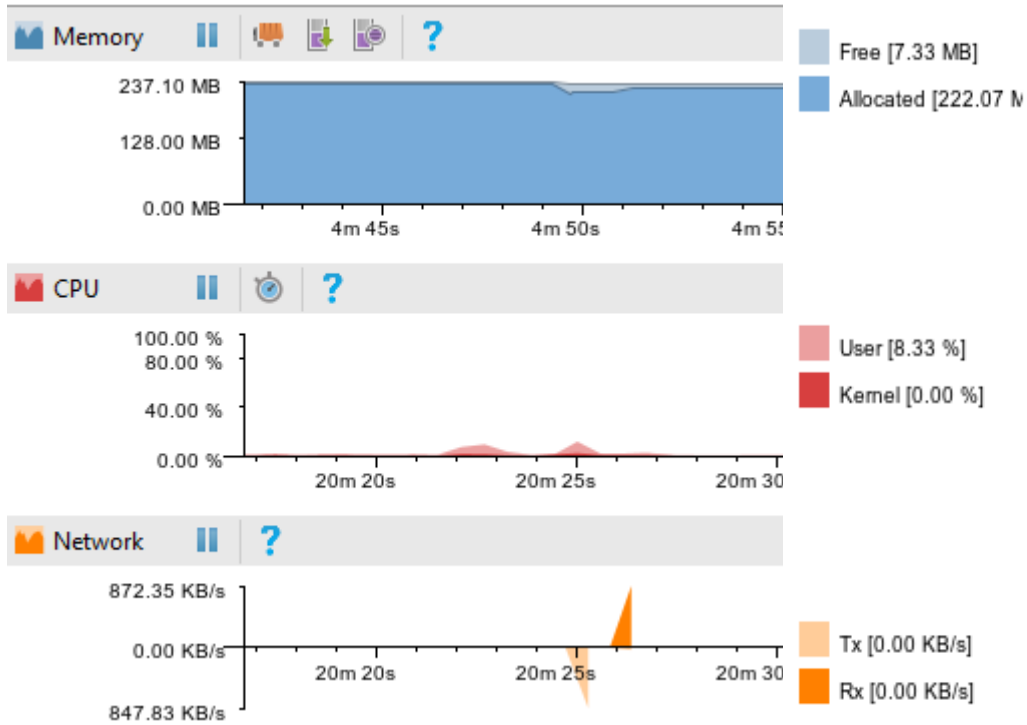
These performance metrics are available through android studio when running the application. Live feedback is provided as user interacts with the system on a connected mobile phone.

It is important to not just have a functioning application but an efficient one. It is not acceptable for CPU usage to rise about 10% of device capability. Network calls should not exceed 10Mb, and device memory consumed by application shall not exceed 250MB. These figures were chosen to give the device the best performance possible without slowing down the users phone.

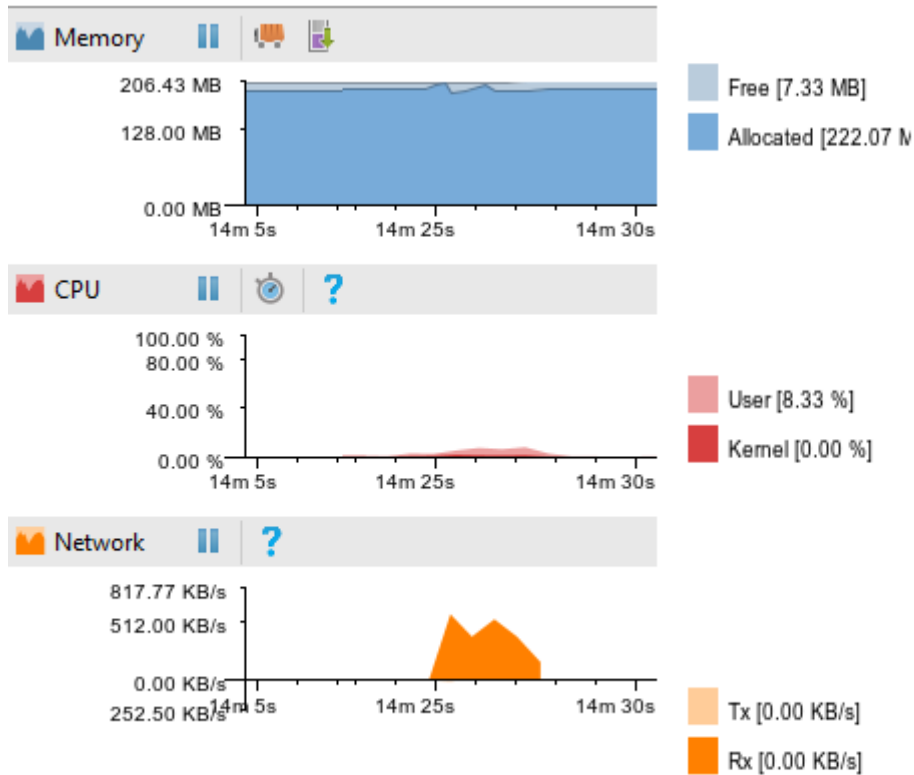
The following image shows system performance indicators during creation of a new bike



The following image shows system performance indicators during querying of the database



The following image shows system performance indicators during member sign on.





The application should be able to facilitate having over 100 active users for the initial release. The backend of the system should be able to handle network requests from all these users simultaneously if such an event should occur.

The database will provide 1GB of free hosting and support over 1000 active connections. These specifications are available with the free version of Firebase, should the application grow in time a premium service may be purchased from Firebase which allows upwards of 5GB to be stored and unlimited connection to DB.

### 3 Conclusions

#### Advantages:

This project has the advantage of using crowdsourcing, as opposed to having one or possible no guards looking for your bike. It gives a user extra tools such as the external sensors to help track down a stolen bike.

#### Limits:

Like any crowd sourced platform this project needs critical mass to make it fully viable. Many users nationwide needed for the application to reach its full potential.

#### Opportunities:

Currently there is no application on the market using smart sensors as a way of theft prevention for bike users.

Also there is no application crowdsourcing the investigative work of returning a user's bike.

Therefore, this application has first-mover advantage in the marketplace, which is critical for success in this field.

## 4 Further development or research

The system has huge potential to expand over time with a growing user base.

We aim to give the user more interactivity with the system. If our user base was to grow significantly, we can expand into overseas markets, offering translated versions of the application.

We could create a “premium” version, where users can have more features for a set monthly price. Potentially only offering the anti-theft sensor feature for a set price, this price would include the sensor being shipped to them.

In the future the application will be made available for all major mobile platforms including android, iOS and windows phone.

The scan for stolen bikes feature of this application would could be a valuable tool for any law enforcement agency. A place car could have an application with this feature running constantly while on patrol similarly with foot patrol police officers. Should they happen to pass a bike or even car which has been listed as stolen and is using this system they will be notified. Through the proximity feature they then can pinpoint the exact location of the vehicle and then decide to investigate further.

I have called into one Garda station and discussed the project with a Sergeant O'Donoavn at Kevin Street station Dublin. During our conversation I demoed my application to the Sergeant and he informed me that there is no such device in use to recover stolen bikes currently and such a technology could have a positive impact on crime solving rates for bike thefts.

I have also contacted bikeshare.ie the company behind the Coca-Cola branded public bikes available in Dublin and other major Irish cities to see if this is technology they would be interested in. unfortunately they have been difficult to get feedback from and I am still awaiting a response from my emails with them.

## 5 References

1. The Irish Times. 2016. Average of 14 bikes stolen a day in Dublin, says cycling group . [ONLINE] Available at: <http://www.irishtimes.com/news/environment/average-of-14-bikes-stolen-a-day-in-dublin-says-cycling-group-1.2397430>. [Accessed 17 October 2016].
2. . 2016. Recorded Crime (Annual) - Datasets. [ONLINE] Available at: <https://data.gov.ie/dataset/recorded-crime-annual>. [Accessed 17 October 2016].
3. Buecheler, Thierry et al. 2010. Crowdsourcing, Open Innovation And Collective Intelligence In The Scientific Method: A Research Agenda And Operational Framework. Department of Informatics, University of Zurich, 2016.
4. "Estimote SDK For Android". Estimote.github.io. N.p., 2016. Web. 3 Dec. 2016.

### In app references

1. BeaconListAdapter.java class adapted from:  
[http://www.programcreek.com/java-api-examples/index.php?source\\_dir=crowdalert-hackdelhi-master/crowdalert-androidclient-main/app/src/main/java/com/roalts/hackdelhiclient/BeaconListAdapter.java](http://www.programcreek.com/java-api-examples/index.php?source_dir=crowdalert-hackdelhi-master/crowdalert-androidclient-main/app/src/main/java/com/roalts/hackdelhiclient/BeaconListAdapter.java)  
original author wiktork@estimote.com (Wiktor Gworek)
2. Icons taken from googles material design icon library  
<https://material.io/icons/>
3. Any other images used taken from free image library  
<https://pixabay.com>

## Gradle Dependencies uses in project:

```
compile 'com.google.maps.android:android-maps-utils:0.4.+'
compile 'com.google.firebase:firebase-storage:10.2.1'
compile 'com.google.firebase:firebase-auth:10.2.1'
compile 'com.google.firebase:firebase-core:10.2.1'
compile 'com.firebase:firebase-client-android:2.5.0'
compile 'com.firebaseui:firebase-ui-storage:1.2.0'
compile 'com.google.firebase:firebase-database:10.2.1'
compile 'com.android.support:appcompat-v7:24.2.1'
compile 'com.android.support:design:24.2.1'
compile 'de.hdodenhof:circleimageview:2.1.0'
compile 'com.android.support:support-v4:24.2.1'
compile 'com.firebaseui:firebase-ui-database:0.4.0'
compile 'com.google.android.gms:play-services:10.2.1'
compile 'com.github.vihtarb:tooltip:0.1.9'
compile 'com.estimate:sdk:0.16.0@aar'
compile 'com.wdullaer:materialdatetimepicker:3.1.1'
compile 'com.codevscolor.materialpreference:mp:0.2.1'
compile 'com.google.android.gms:play-services-places:10.2.1'
compile 'com.google.guava:guava:20.0'
compile 'com.google.android.gms:play-services-location:10.2.1'
```

## 6 Appendix

What Follows Is a range of supporting documents including the project proposal, project plan and key terms explained.

### 6.1 Key Terms

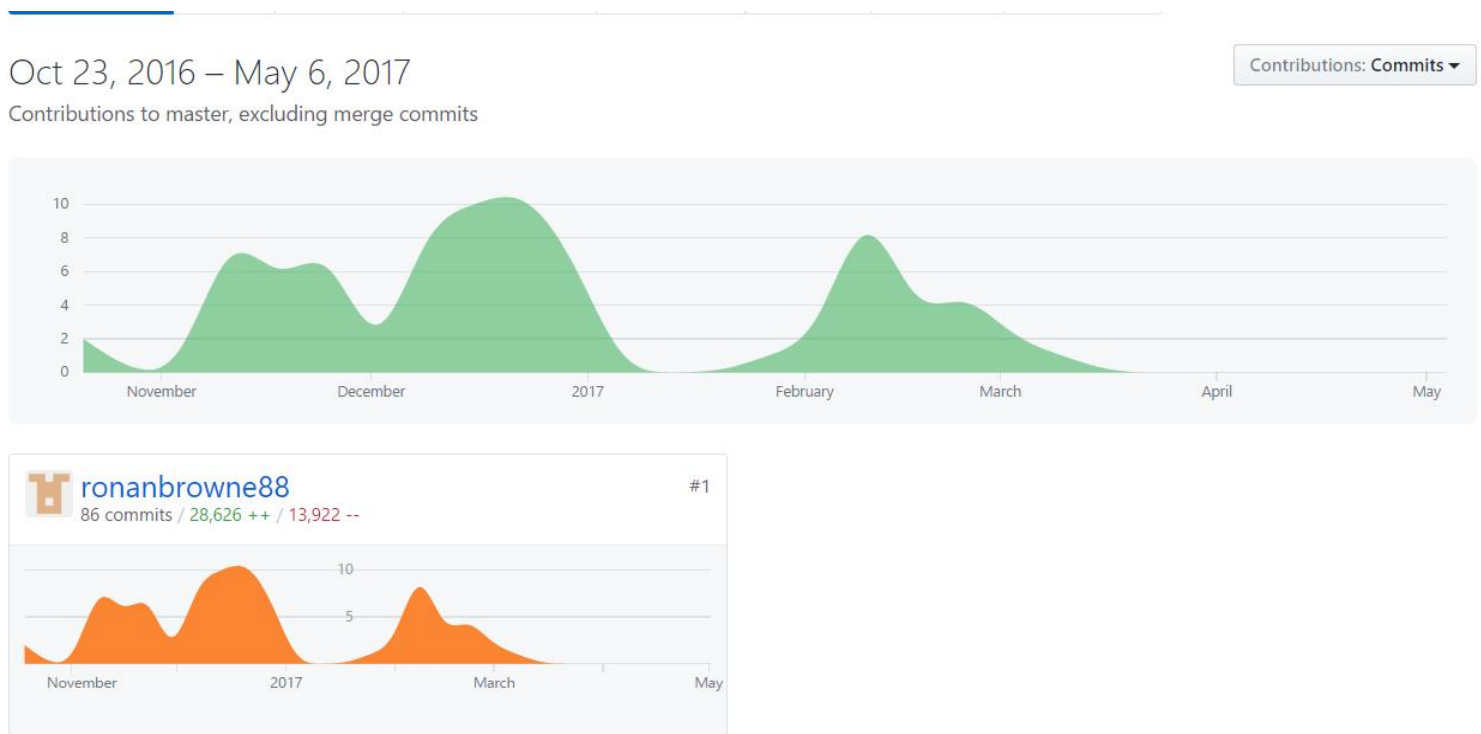
The following table provides definitions for terms relevant to this document.

Term	Definition
Gradle	A build tool used to build packages and manage dependencies. 3 <sup>rd</sup> part libraries can be added to a project by adding their dependency to the grade file. Gradle will then download the necessary components to use the classes and library's associated with this dependency.
API	Application Programming Interface this is a set of programming instructions and standards for accessing a Web-based software application or Web tool.
Beacon or Sensor	These terms may be used interchangeably throughout the documentation, they refer to the external hardware devices used in this project which broadcast a signal that is picked up by the users phone to establish a link between a bike and user.
Firebase	Cloud based developer platform provided by Google, provides database functionality for project.
DB	Refers to the systems database
SSL	Secure Sockets Layer is the standard security technology for establishing an encrypted link between a web server and a client. This link ensures that all data passed between the web server and client remain private and integral.
Client	An application which receives information from a server. In this case a client in the application running on a user's phone.

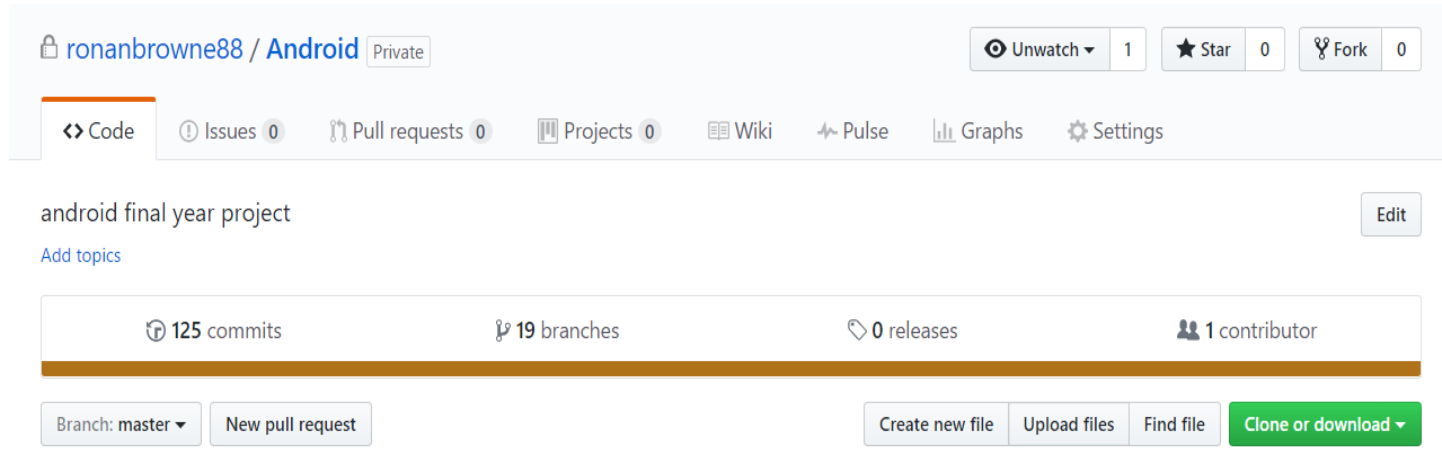
UI	User Interface, how the user will interact with the system.
Loose Coupling.	Measure of dependency between two software modules.
Dependency	Dependencies means the things that support the build of your project such as required JAR file from other projects and external JARs such as JDBC

## 6.2 Project progress tracked through GIT.

Below we have a screenshot showing the applications coding progress as tracked by GitHub. GitHub was used as the version control repository for this project.



Below we have a screenshot of my private Git repository which hosts this project. You can see at the time of submitting this document the project has had 125 commits and 19 branches.



### 6.3 Project Proposal

The original documentation submitted as the project proposal follows.





# PROJECT PROPOSAL

Summary

Ronan Browne  
x13114522  
[x13114522@student.ncirl.ie](mailto:x13114522@student.ncirl.ie)

### **6.3.1 Objectives**

The aim of this project is to develop a scalable, community driven anti-theft focused mobile application. This should be a streamlined responsive application regardless of number of users and provide an intuitive user experience with a natural flow between screens.

Scalability will also be kept in mind regarding the product this application is aimed to protect. Initially this project will be focused on bikes but further iterations of the application could easily include cars etc.

This application should be community driven with a strong emphasis on user interaction, the users will populate the application data. The application will facilitate users to help one another locate their stolen bikes

A user will have a unique identity on the app, they can then associate one or multiple bikes with this user ID, providing all identifying features of the bike including a photo. In the event a user's bike stolen they then update their profile to reflect this, bike is then uploaded into our "stolen DB" which can be viewed by any user of app. A user can then interact with other bikes they see in DB reporting if they have seen it somewhere. Alerting authorities and original owner.

This app will also provide google maps integration to allow users to drop a marker on an area where their bike was stolen, this map data will be visible to all users thus over time we will see theft hotspots develop and users can use this data to help make better decisions on where to leave their bike parked.

Overtime a large DB of stolen bikes could be built up so it is import that the application allows the users to provide excellent queries to this data. Narrowing down fields such as location and bike models when searching through bikes that have been reported stolen.

The target user for this application is any bike owner with an android smartphone, initially we will focus on Dublin City centre as our target area for the beta release of this application.

During my project pitch one of the suggestion from my panel for this project was implementing an Internet of things aspect to this project. This is something I will assess as I build my, due to scope and resources required to implement this at this stage it is undecided if it will be implemented in the final project. I will discuss this with my supervisor in the weeks to come.

### **6.3.2 Background**

#### **Inception**

With an average of 14 bikes stolen every day in Dublin (Dublin Cycling Campaign) this is a real issue for cyclists. More often than not the bike is never seen again.

I first thought of the concept for this idea several months ago when my own bike was stolen in Dublin city center. Had an application like this existed it would have been an extra tool which may have ultimately led to me retrieving my bike.

## **Research**

After coming up with the idea I did research on the Play store to see what is currently available similar to my idea. I found an app called “check my bike “(link in references) which provides tips on how to prevent bike theft and a feature to register your bike. The application is poorly designed. Each menu option simply launches a web page in your browser to their site and some of the links are dead. This is not a fully mobile solution, simply an app which launches links to their web site.

Another app I found during my research was Bike Theft Alpha, which claims to provide some features similar to mine (Bike registration / maps implementation), Although the app is on the play store it seems to be an early release as it currently stands on October 18<sup>th</sup> none of the key features in this application actually work, Registered bikes don't display, maps feature not working, other application screens are blank and it is listed at having less than ten downloads.

Also During my research, I found no apps which are community driven and provide a tool for reporting sightings and google maps integration.

Crime statistics hosted on data.gov.ie do not provide specific data on bike crime so I feel there is a gap in the market for my application. Especially the feature which will allow users to identify theft hot spots on a google maps overlay.

### **6.3.3 Approach**

#### **Challenging features**

One of the challenging aspects of this project is to come up with a back end solution for this application, which will remotely sync and keep user's info updated in real time. After researching various cloud back end tools for android applications (Parse, Firebase).

I decided to go with google Firebase for a number of reasons. Including easy integration into android studio and the documentation is thorough and it is supported by a large community of developers. This is not a technology I've used before so it will be self-learned during the process of developing this project.

On the bike registration screen instead of a user entering a list of bike details in a table fashion I will make the app more interactive by having an image of a bike which a user can click on various parts, such as wheels or frame to enter specific details. Making particular areas of the one image clickable is something I will have to research as I have not previously implemented this.

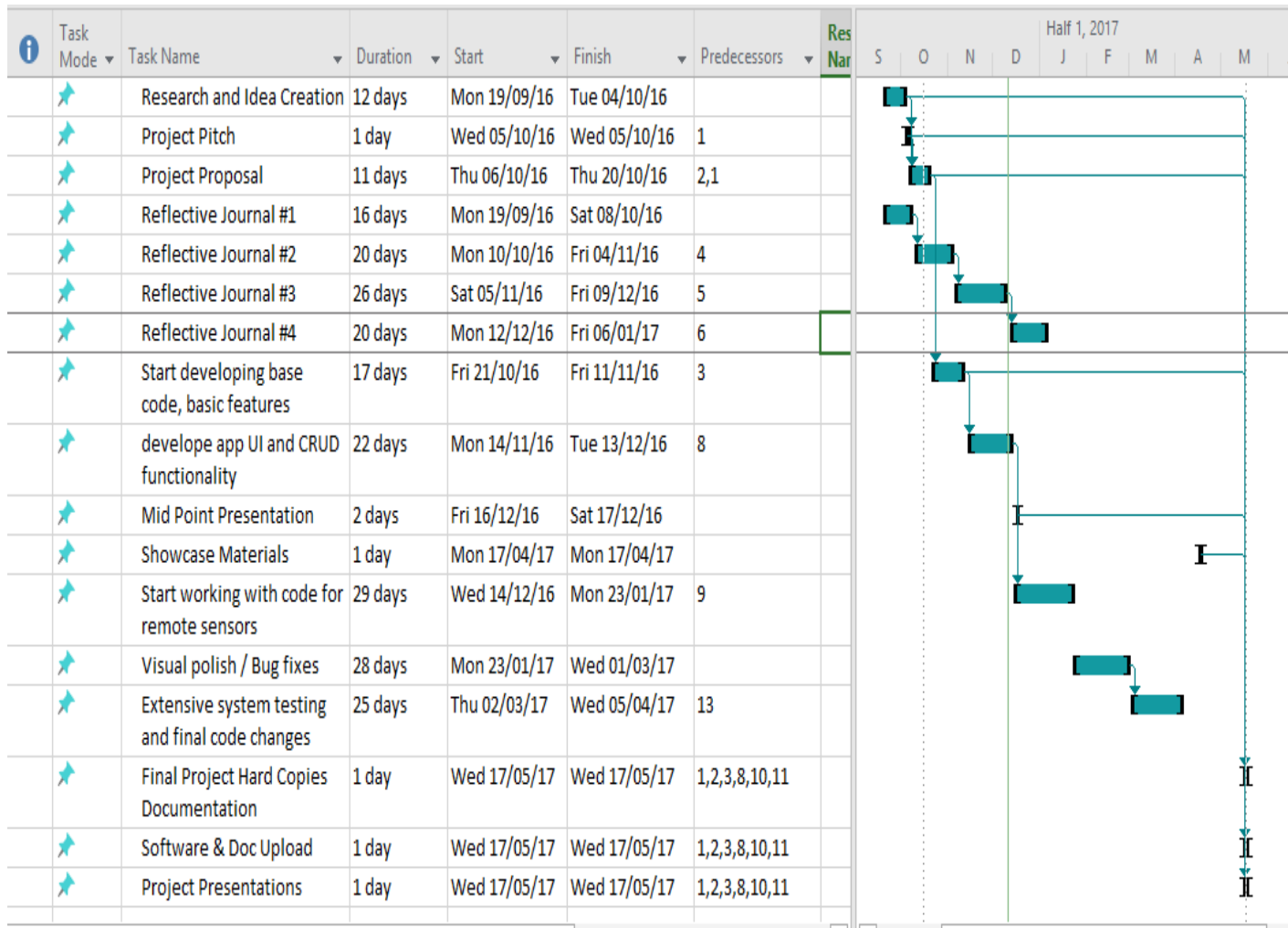
Firestore will allow me to provide a real time, scalable backend to my application. It is a Non SQL database with uses JSON to structure its data.

Google maps integration into this project will also be a challenge as it is something I have no previous experience with.

After initial creation of wireframes, my first step will be to create a basic UI in Android Studio where I can prototype some of my key features which will be showcased in the mid-point presentation.

#### **6.3.4 Project Plan**

This Gantt chart is set up to track the project deliverables throughout the year. This will be a very useful for me when it comes to time management.



### 6.3.5 Technical Details

#### IDE and language

This project will be developed using Android Studio 2.2.1. The application will be developed solely for the android platform using the java programming language. It will be designed for use on phones running SDK 15 (Ice cream sandwich) and above (compatible with 97% of all android devices)

#### 3rd party libraries

The 3rd party libraries used will be google firebase and Espresso UI Testing. Firebase will handle the data storage needs and espresso is a framework for automated user Interface testing on Android.

## Version Control

I will be using version control tool GIT while developing this project. Code will be stored online in a remote repository.

### 6.3.6 Evaluation

Testing is a key part of the development life cycle. My project will make use of Unit tests to test any non UI code and also automated UI testing using the Espresso framework. On top of this testing rigorous exploratory testing will be done on my personal device as features are being developed.

My main device is a Samsung S6 Edge+ , But to ensure the application runs on a variety of devices I will use android studios inbuilt emulator to test the application on a range of different powered and sized devices.

Once development reaches a stage where the main functionality of the application is usable i will get several end users to test the application. These end users will be my peers. I will instruct the, to use the app as if it were an official release. I will instruct these users to report any bugs or unusual behavior to me.

Ronan Browne 17/10/2016

\_\_\_\_\_  
Signature of student and date

### 6.3.7 References

5. AndroidBegin. 2016. List of the best Android Backend as a Service (BaaS) provider - AndroidBegin. [ONLINE] Available at: <http://www.androidbegin.com/blog/list-best-android-backend-service-baaS-provider/>. [Accessed 17 October 2016].
6. The Irish Times. 2016. Average of 14 bikes stolen a day in Dublin, says cycling group . [ONLINE] Available at:

<http://www.irishtimes.com/news/environment/average-of-14-bikes-stolen-a-day-in-dublin-says-cycling-group-1.2397430>. [Accessed 17 October 2016].

7. Check That Bike! - Android Apps on Google Play. 2016. Check That Bike! - Android Apps on Google Play. [ONLINE] Available at: <https://play.google.com/store/apps/details?id=com.cooper.checkthatbike&hl=en>. [Accessed 17 October 2016].
8. . 2016. Recorded Crime (Annual) - Datasets. [ONLINE] Available at: <https://data.gov.ie/dataset/recorded-crime-annual>. [Accessed 17 October 2016].
9. BikeAlpha Bike Theft Reporting - Android Apps on Google Play. 2016. BikeAlpha Bike Theft Reporting - Android Apps on Google Play. [ONLINE] Available at: <https://play.google.com/store/apps/details?id=com.ennisoft.MyBikeAlpha&hl=en>. [Accessed 18 October 2016].

## **6.4 Monthly Journals**

### **6.4.1 Reflective Journal Month 1**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **September** Upload one journal every month. Expected word count 300 words (of your own words).

#### **My Achievements**

This month I returned to college after my summer work placement. It was slightly daunting returning to 4th year as I knew there would be a lot more pressure on us this year.

However, I felt able for this given my new knowledge gained during the summer months working for a software company.

This month I gave a lot of thought to my project, upon starting the year I was not sure what I would pick for my final project so I had a lot to think about. After much deliberation I settled on a mobile application which would be a community driven anti-theft solution. This solution is targeted at bicycles but could easily be applied to a wide range of things. The high level overview of this project is that a user may register a bike, report it stolen, it is then uploaded to a "Stolen bike DB" which is accessible by all users of the app. This DB is presented in a user friendly way where a user can report sightings of a particular bike. And also mark areas of theft hotspots on a Google Maps interface.



My contributions to the projects included, gathering my ideas and presenting my project pitch to a panel. My project was approved with some revisions suggested.

### **My Reflection**

I felt, it worked well to brainstorm when coming up with my initial idea. Also drawing from real life experience helped me come up with this idea. My own bike was stolen last year so a tool like this could have helped me see it returned.

While it felt good to have the project approved after the pitch, the pressure of the year ahead and implementing all these features is daunting. Managing my time between this and all the other course work is going to be a big issue.

### **Intended Changes**

Next month, I will try to start implementing the basic prototype of my project and start writing up the specifications documentation.

I realized that I need to meet with supervisor for further guidance. Refine my ideas more and pin down the exact scope for this project.

### **Supervisor Meetings**

Date of Meeting: \* **Supervisor not yet assigned.**

Items discussed:

Action Items:

## **6.4.2 Reflective Journal Month 2**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **October** Upload one journal every month. Expected word count 300 words (of you own words).

### **My Achievement**

This month the major mile stone was completing my project proposal which I have already submitted. This is a living document some aspects of it may change as I communicate with my supervisor and decide what is the best approach for my project.

After being back to college for a month now I feel I have successfully settled into academic life and despite initial worries about time management I'm feeling more confident about this year.

Also during this month, I had my first supervisor meeting. My supervisor is Paul Hayes. We discussed my project direction as well as some action points for the coming month.

I also created a Git Hub account and created a repo for my project there. This will be useful for version control. I started to implement a basic code base for my project in Android studio also.

<https://github.com/ronanbrowne88/Final-Year-Project>

### **My Reflection**

I was initially hesitant about implementing a IoT project into this as suggested by the review panel who approved the project. I was afraid the scope would be too large and the cost of the needed hardware too great. However, after further evaluation and reflection I have decided this would be a worthwhile feature to develop. As it will help my project stand out from others both in the college and commercially.

### **Intended Changes**

Next month I will be showcasing a basic prototype so I will continue to work on the code base for this, using GIT for version control. I hope to have a bare bones

prototype of some of the main features by then. Such as registering a bike, viewing stolen bike DB. Editing a registered bike. As I still have to research and purchase the hardware used for the Bluetooth enabled proximity sensors I will be using this element may not make it into my first prototype showcase.

## **Supervisor Meetings**

Date of Meeting: **25/10/2016**

Items discussed: Project direction, project scope and timeline. Inclusion of internet of things aspect of project.

Action Items: start prototyping, start requirements, keep up to date with reflective journals.

### **6.4.3 Reflective Journal Month 3**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **November** Upload one journal every month. Expected word count 300 words (of you own words).

## **My Achievement**

This month I continued my work on the project code base. Expanding several key areas. Including user profile features. Querying my database, implementing google maps and a sliding navigation drawer interface.

I have used Git extensively during this phase of development, regularly creating new branches to work on isolated features. Since my last Journal I've made of 20 commits to my repository.

I implemented several APIs and third party library's into my project so far. Including the google maps API and smaller third party library's such as CircleImageView

(<https://github.com/hdodenhof/CircleImageView>) . the latter is a library which allows a developer to shape an Image into a custom circular appearance. Im using this to display a user's profile the application.

This month I ordered and received the Bluetooth sensors I will be using as part of this project (<http://estimote.com/>) . I have not had a chance to start testing this feature yet. I feel the sensors will be a time consuming task so I want to polish off other sections of the application before I get start diving into integrating the sensors into my project.

I also continued my meetings with my supervisor Paul Hayes during this time who seemed happy with the progress and direction of my project thus far.

### **My Reflection**

After using GIT for more than a month I can see how it is a useful tool for development. GIT also has GUI integration with Android Studio my IDE of choice for this project.

I feel I have quit a lot of the projects base code implemented at this stage. This being a project I am interested in and have a passion for, I have found it hard to restrain myself from the coding aspect of it until later in the year as per Eamons recommendations. I have on any occasions found myself writing code well after midnight. Possibly to the detriment of my other subjects. Going forward I will have to re-evaluate my time management to make sure my other subjects are getting enough attention.

### **Intended Changes**

This month I shall have my prototype demonstration, I may have feedback from that which I will implement in my project. For now, I shall continue working on the current features. In the next few weeks I hope to start implementing the code which will allow the Bluetooth sensors to communicate with the application

## **Supervisor Meetings**

Date of Meeting: **22/11/2016**

Items discussed: project direction, midpoint presentation, Integration of Bluetooth sensors to give this project an IOT angle.

Action Items: continue work on prototyping, keep journals up to date, prepare for mid-point presentation.

Date of Meeting: **02/12/2016**

Items discussed: midpoint presentation plan, advice on technical documentation.

Action Items: continue prototype work, focus on completing technical documentation for mid-point presentation.

### **6.4.4 Reflective Journal Month 4**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **December** Upload one journal every month. Expected word count 300 words (of your own words).

#### **My Achievement**

This month I had my midpoint presentation with Eamon Nolan and Paul Hayes. I did a lot of preparation this month in the lead up to this. This included practicing my presentation skills and practicing demonstrating my application.

The presentation went well and I achieved a high grade, 23% out of a potential 25. I was happy to get recognition for all the hard work I've put into the project thus far. This also motivated me to keep working hard towards the final presentation.

This month fell during Christmas and the first semester college exams so project work has taken a back seat for these few weeks as I have been mainly focusing on my exams. And with this I have not had a chance to meet with my supervisor since the mid-point presentation.

I hope to meet with Paul Hayes this week so I can receive feedback and we can discuss implementing next steps of my project.

While I have mainly been focusing on exams, I have had a chance to do some project work, the changes made this month include, Refining the query system for database searches a user can now search by date, i.e. show me only bikes stolen in last month.

I made many small GUI changes which I feel improve the overall look and feel of the project. Such as changing button colors, styling and placement of some widgets.

I also started experimenting with the Bluetooth sensors which I bought last month. I created a small side project just to try and connect a sensor to my device and get a reading back.

### **My Reflection**

After some experimentation with my external sensors im confident with some work I can achieve the functionality set out in my technical document. (notification of unauthorized movement)

The holidays and exams this month have seen a slight drop in time I could devote to the project. While this has hindered practical development, I feel stepping away from the code for a while and coming back fresh may give me new perspective on solving any technical issues faced

### **Intended Changes**

Implement any changes suggested by my supervisor based on midpoint presentation.

Continue to work on the code base refining UI as needed.

Start to implement the Bluetooth sensors into the project. And have the application successfully detecting a bikes (represented by a sensor) movement in and out of range.

### **Supervisor Meetings**

14/12/2016 – midpoint presentation was my main contact with supervisor this month due to holidays and conflicting time schedules.

## **6.4.5 Reflective Journal Month 5**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **January**

### **My Achievement**

This month seen the return to regular class after the exams and Christmas break, with this return to structured college I was able to re-focus my time and energy on this project.

I was able to expand on the functionality of the Bluetooth sensors I first began working with seriously last Month.

I programmatically used Near Field Communication(NFC) with the sensors and my device. If a user has a NFC enabled device they can tap their phone off a sensor and it will launch the application, if the application if not installed it will launch the Google play store link where a user can download the application.

I was also able to successfully link a phone and the external sensors in my application, the system now detects if a bike is in range (represented by a sensor) and the user receives a notification if the sensor leaves range.

### **My Reflection**

Im happy I was able to implement the sensors successfully into my project.

It's rewarding and motivating to see key areas of the system start to take shape and work as intended.

The holidays and exams this month have seen a slight drop in time I could devote to the project. While this has hindered practical development, I feel stepping away from the code for a while and coming back fresh may give me new perspective on solving any technical issues faced



## **Intended Changes**

Continue to work on the code pertaining to the external sensors. While the basic idea is currently working it needs some refining, there is currently a thirty second wait time for the phone to recognize a break in signal from the sensors I would like to get this lower as I feel it would be more practical for the purposes of my app, however after some initial research this time contrite may be a hardcoded software configuration set by the manufacturers. It is something I will research more this month.

Furthermore, I will begin to implement more advanced sensor code such as actively scanning an area for the sensors of stolen bikes.

Also I will continue working on the documentation and start planning the various testing that will need to be done on this project.

## **Supervisor Meetings**

2/2/2017 – I Met with Paul Hayes to discuss feedback from the midpoint presentation, he was over all happy with how thing went. We discussed possible other uses for the system such as theft prevention for other objects such as cars. This is something I shall look into more.

I also demonstrated the new functionality of the application including sensor capability's and NFC launching of app.

## **6.4.6 Reflective Journal Month 6**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **February**

### **My Achievement**

I continue work on my final year project this month, at this stage all major features have been implemented and I spent a lot of this month making minor tweaks to the code to improve performance and fix any bugs, along with minor user interface changes, such as implementing loading bars and help areas to give more feedback to the user.

Also, this month I began coding test classes. Using the Junit test framework. Testing is an important aspect of this project so this is something I will continue to work on for the remaining weeks.

### **My Reflection**

With the Semesters end now approaching I'm starting to feel both anxious and excited to finish this project. I believe I should be on schedule to have all features within my intended scope operational for the upcoming showcase.

### **Intended Changes**

Continue to thoroughly test the application, ensuring there are no hidden bugs. Focus on completing all elements of the documentation and complete the showcase profile which Eamon has instructed us to do.

Also, this month I will work on making sure my application is fully usable and assessable to a wide range of people by implementing options for making text

bigger, changing the color scheme (certain colors may suit color blind users more) or using the application in languages other than English.

### **Supervisor Meetings**

3/3/2017 – I Met with Paul Hayes to discuss the progress of my application this far, He was pleased with the progress and encouraged me to keep working hard in this final stretch.

#### **6.4.7 Reflective Journal Month 7**

Student name: **Ronan Browne**

Program: **BSc in Computing, Mobile Stream**

Month: **March**

#### **My Achievement**

Continued to work on some non-visual aspects of my project this month such as tasting and documentation, I also completed my profile for the showcase this month.

#### **My Reflection**

This month say a lot of preparation for the end of year showcase, I lot of pressure was put on us to complete or profiles, upload photos and get them signed off all of which I did. As the year is nearing a close a lot of modules have deadlines now so it has been hard to assign time to my main project, thankfully I put in a lot of hours in the early months and have completed all major features before this point had I not started coding till after Christmas like some of my peers I would be under substantially more pressure now.

## **Intended Changes**

The next month will see me coming into the final stretch of this project so a lot of time will be spent on preparation for the final showcase and final presentation, all application features especially some of those developed months ago must be re-examined and tested to ensure correct behavior when the final deadline comes.

Specifically, I need to continue writing code for unit tests and with test cases for usability testing. A large portion of time will be spent reviewing and editing the final technical report also.

## **Supervisor Meetings**

20/3/2017 – I dropped into DR. Hayes to update him on the progress of my project. Since I had only minor visual updates since our last meeting there was not much to show and I just assured him I was continuing to work on the project and would contact him if I needed any assistance.